第 1 章　SCILAB 简介

1.1　引言

1.2　SCILAB 软件的构成

1.3　安装 SCILAB 的系统需求

1. 4　SCILAB 主窗口介绍

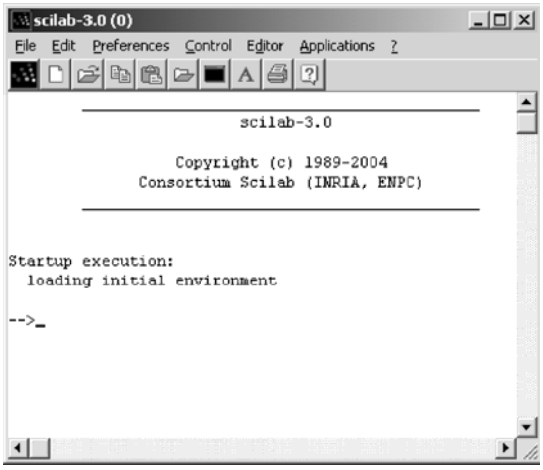

图 1.1　SCILAB 3.0 的主窗口

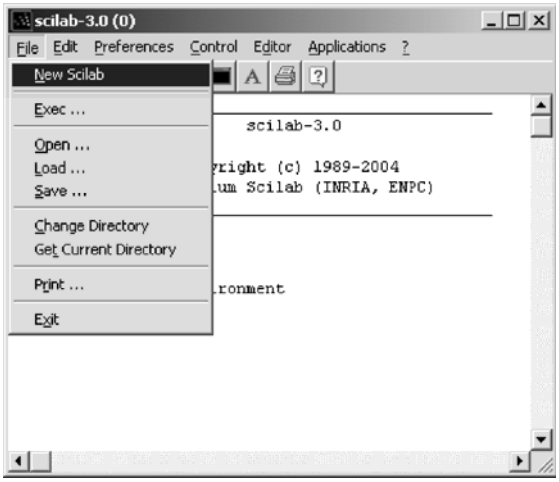

图 1.2　SCILAB 3.0 的文件菜单项

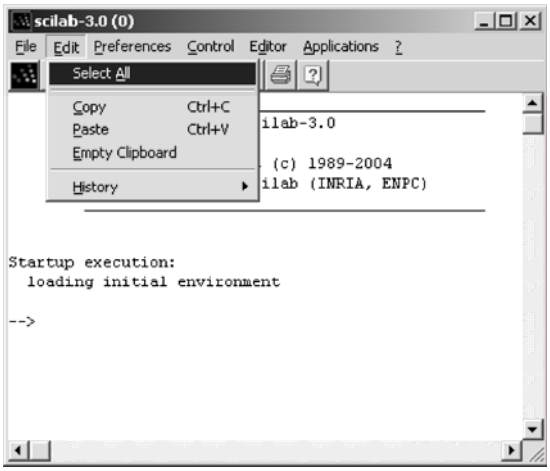

图 1.3　SCILAB 3.0 的编辑菜单项

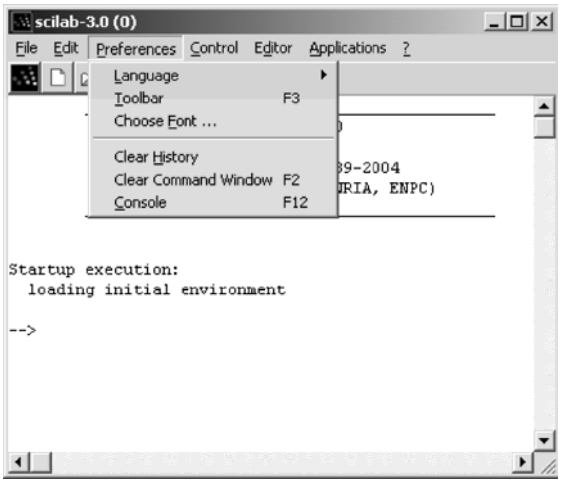

图 1.4　SCILAB 的选择菜单项

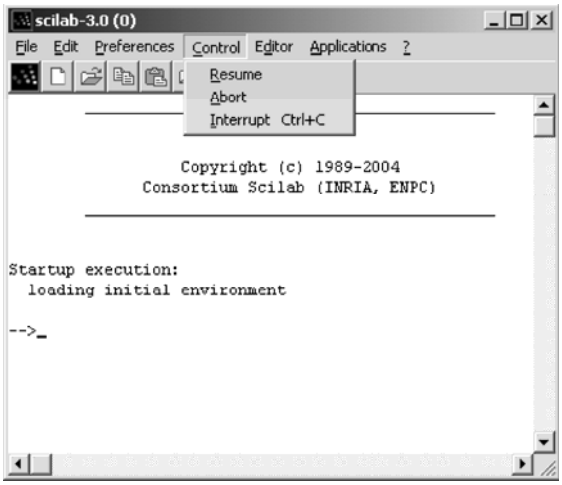

图 1.5　SCILAB 的控制菜单项

图 1.6　SCILAB 的编辑器窗口

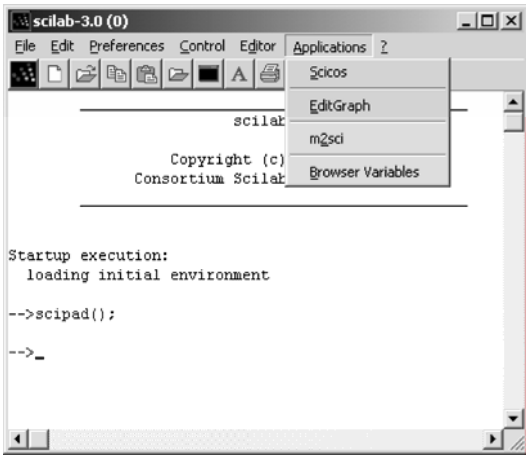

图 1.7　SCILAB 的应用窗口

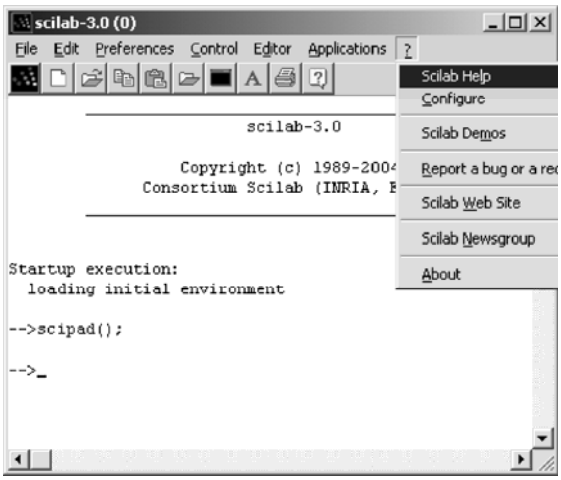

图 1.8　SCILAB 的帮助窗口

1.5　SCILAB 中的基本操作与预定对象

1.5.1　SCILAB 中的文件操作

```
-->getcwd()
ans =
C:\Windows\Desktop
```

也可以用指令形式显示当前目录

```
--> pwd
ans =
C:\Windows\Desktop
```

1. 5. 2　界面层次的控制操作

```
--> a=5
a =
 !5!
--> b=6
```

```
b =
 !6!
--> pause
-1->c=a+b
   c =
     !30!
-1--> resume
--> d=a+c
!--error  4  !
undefined variable : c
```

```
a =
!5!
```

```
b =
 !6!
```

1.5.3　SCILAB 主窗口中的快捷键操作

表 1.1　快捷键的操作功能说明

键　名	操 作 功 能	键　名	操 作 功 能
↑	向前搜寻已经输入的指令	Delete	删除当前光标字符
↓	向后搜寻已经输入的指令	Esc	清除指令行的全部内容
←	光标向前移动一个字符	Ctrl＋c	指令界面上升一层
→	光标向后移动一个字符	Ctrl＋p	向前搜寻已经输入的指令
Home	使光标移到指令行首端	Ctrl＋n	向后搜寻已经输入的指令
End	使光标移到指令行尾端	Ctrl＋b	光标向前移动一个字符
Backspace	删除光标左边字符	Ctrl＋f	光标向后移动一个字符
Ctrl＋a	使光标移到指令行首端	Ctrl＋u	清除指令行的全部内容
Ctrl＋e	使光标移到指令行尾端	Ctrl＋k	删除当前光标及其右边的字符
Ctrl＋h	删除光标左边字符	Ctrl＋w	删除指令行的最后一个词
Ctrl＋d	删除当前光标字符		

1.5.4　SCILAB 中预先定义的对象

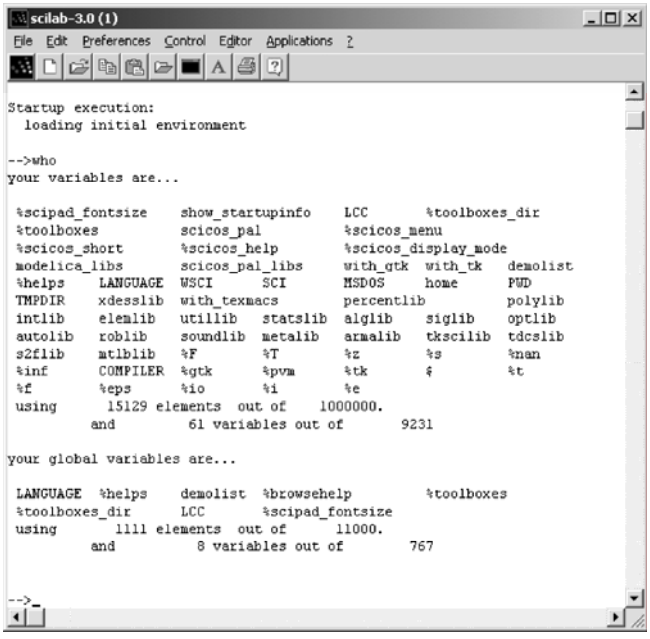

图 1.9　who 指令后的 SCILAB 主窗口

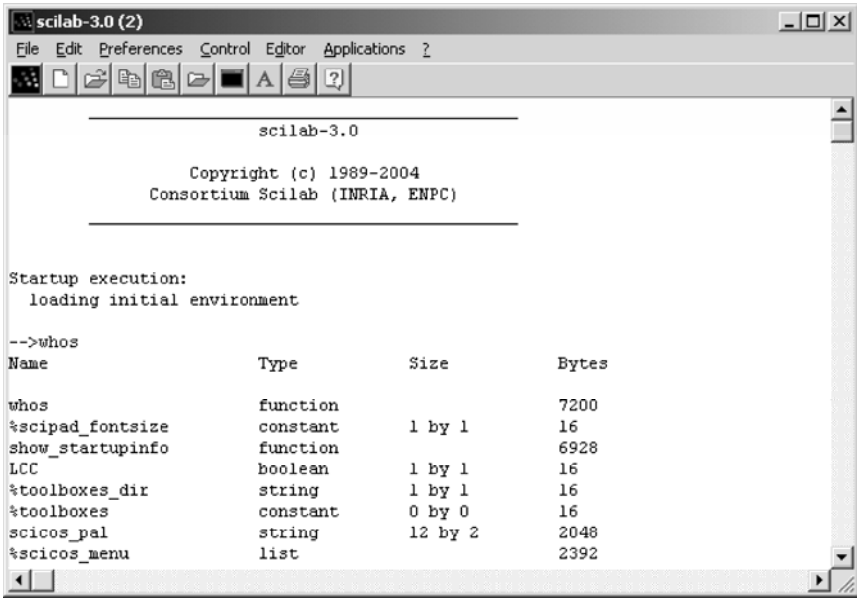

图 1.10　whos 指令后的 SCILAB 主窗口

```
-->typeof(%i)
ans =
    constant
```

1.6　谈谈如何学习 SCILAB

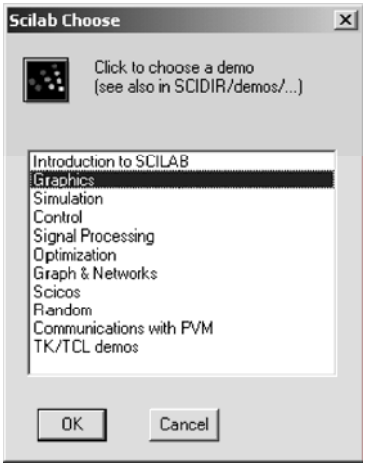

图 1.11　SCILAB 的演示窗口

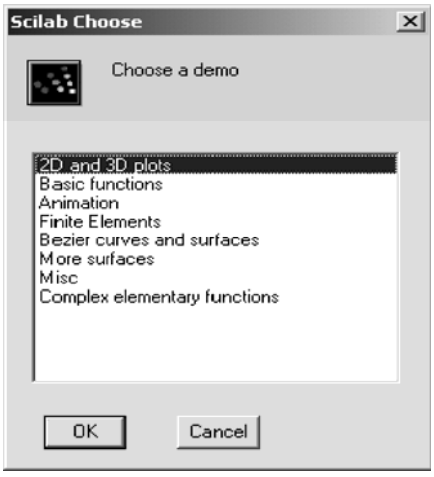

图 1.12　SCILAB 的选择图形演示窗口

图 1.13　SCILAB 的选择图形演示窗口

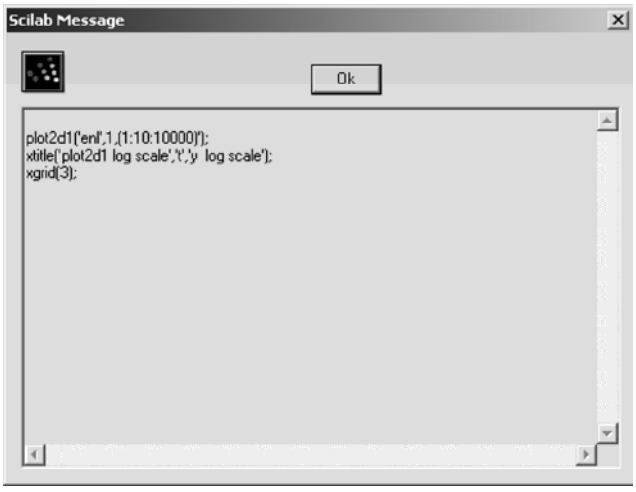

图 1.14　SCILAB 的一个图形信息窗口

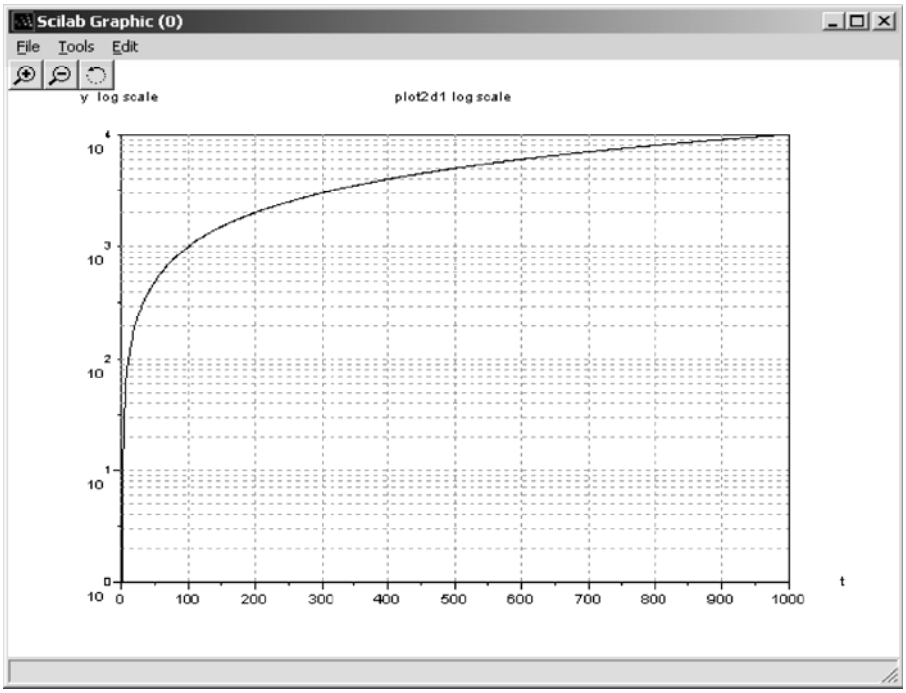

图 1.15　SCILAB 的一个图形演示窗口

第 2 章　数据类型

2.1　引言

2.2　特定符号与常数

2.2.1　特定符号

```
→ a = 3.14 * 2,   b = 3.14 * 3

a =
 ! 6.28 !
b =
 ! 9.42 !

→ a = 3.14 * 2;   b = 3.14 * 3

b =
! 9.42!

→ 变量名
```

```
→a ↙
```

```
→ a = 2 * 3.1415926 * R+ …
→ 3 * 5.692 * sin(8.27) + …
      → 9.8 * 3.6 * 2  ↙
```

```
      → a = 3; b = 4; ↙
      → a * a + b * b ↙
       ans = 25
      →   // Now we have created variables and list them by typing:
       Who ↙
your variables are …
ans   b         a   …
…
```

2.2.2 特定常数

```
→ a = sqrt( - 1)
```

```
→(2 + %i) * (2 - %i)  ↙
 ans =
  ! 5 !
```

```
→ 2^8888  ↙
  ans = Inf
```

```
   →2/(2 - 2.0)  ↙
! _ _ error
division by zero...
```

而 nan 指的是“不是一个数”或者可以理解成“不能确定的数”，例如

```
→ 2 ^88888 — 3^66666 ↙
     ans =
     nan
```

2. 3 标量的数值计算

```
   → 2 + 3 ↙
      ans — 5
```

```
   → 2 — 8 ↙
      ans — — 6
```

```
→ 3 * 8 ↙
   ans=24

→ 4/2 ↙
   ans =
      ! 2 !

→ 2\4 ↙
   ans =
      ! 2 !

→ 2^4 ↙
   ans =
      ! 16 !
```

2.4 数值型向量与矩阵的定义及基本运算

2.4.1 数值型向量与矩阵的定义

```
→ x=[0.5,2.0,3.8,4.9] ↙
x =
   ! 0.5  2.0  3.8  4.9 !

→ x=[ 1:0.5:3] ↙
x =
   ! 1.0  1.5  2.0  2.5  3.0 !

→ x=[ 1:5 ] ↙
x =
   ! 1  2  3  4  5 !

→ x=1:0.5:3 ↙

→ x=1:5 ↙

→ x=[0.5:0.8:1.9:1:6] ↙
  x =
  ! 0.5  0.8  1.9  1  2  3  4  5  6 !

y=linspace(d1,d2 [,n])

y=logspace(d1,d2 [,n])
```

```
A. → y=linspace(1,10,5)↙
     y=
     !   1.   3.25  5.5  7.75  10  !
B. → x=logspace(1,2,5)↙
     x=
     !   10.   17.782794  31.622777  56.234133  100.  !
```

```
→ x=[0.8,0.96␣ 0.32␣ 1;7,36␣ 280␣…
      53,2,8␣ 9;0.5;11␣ 22.9];  ↙
```

```
→ x=[ 3.6; 7.8; 2.9 ]  ↙
  x=
    ! 3.6 !
    ! 7.5 !
    ! 2.9 !
```

```
→ A=[3,4.5,5; 1;3; 2␣4␣6.5]  ↙
  A=
    ! 3  4.5  5 !
    ! 1  2    3 !
    ! 2  4  6.5!
```

2.4.2 数值型向量与矩阵的运算

```
→ a=[3.5,4.2,9.6];  ↙
→ b=[2.7,8.6,1.7];  ↙
→ x=a+b↙
  x=
    ! 6.2  12.8  11.3 !
```

```
→ a=[1,2; 3,4];  ↙
→ b=[5,6; 7,8];  ↙
→ x=a-b↙
  x=
  ! -4  -4 !
  ! -4  -4 !
```

```
→ a=1:3;  ↙
→ a'  ↙
a' =
  ! 1 !
  ! 2 !
  ! 3 !
→ b=[1+%i,2-%i];  ↙
→ b'  ↙
b' =
  ! 1-i !
  ! 2+i !
→ a=[1:3; 4:6];  ↙
→ a'  ↙
  a' =
    ! 1  4 !
    ! 2  5 !
    ! 3  6 !
```

$$c_{ij} = \sum_{k=1}^{P} a_{ik} b_{kj}$$

例 2.4

```
→ a=[1,2,3; 4,5,6];  ↙
→ b=[1,2; 3,4; 5,6];↙
→ c=a*b↙
c =
   ! 22  28 !
   ! 49  64 !
```

例 2.5

```
→ a=[2,1; 2,5];  ↙
→ b=[1,2; 3,6];↙
→ c=a\b↙
c =
   ! 0.25  0.5 !
   ! 0.5    1  !
→ d=b/a↙
  d =
     ! 0.125  0.375 !
     ! 0.375  1.125 !
```

例 2.6

```
→ a=[8,2; 1,5];  ↙
→ b=[3; 2];↙
→ x=a*b↙
x =
   ! 28 !
   ! 13 !
→ y=b'*a↙
y =
   ! 26  16 !
```

$$\boldsymbol{A}\otimes\boldsymbol{B}=\begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2n}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix}$$

例 2.7

```
→ a=[2,4; 1,3]; ↙
→ b=[1,0; 0,1]; ↙
→ c=a.*.b ↙
c=
   !2  0  4  0!
   !0  2  0  4!
   !1  0  3  0!
   !0  1  0  3!
```

例 2.8

```
→ A=[2,1; 1,4]; ↙
→ C=A^2  ↙
C=
   !5   6!
   !6  17!
```

例 2.9

```
→ A=[2,1; 1,4]; ↙
→ C=A^0.5  ↙
C =
  !1.3825476    0.2975940!
  !0.2975940    1.9777355!
```

例 2.10

```
→ A=[0.5,1.5; 2,3]; ↙
→ b=[2,4; 0.5,1.5]; ↙
→ C=A.*B ↙
C=
   !1.0  6.0!
   !1.0  4.5!
```

例 2.11

```
→ A=[1,2; 2,4]; ↙
→ B=[1,0; 0,1]; ↙
→ C=B./A  ↙      //所得为与 A、B 同维数的数组，其元素 c_ij = b_ij / a_ij
C=
   !1    0  !
   !0  0.25!
```

例 2.12

```
→ B=[1,1; 1,1];↙
→ D=A.\B ↙        //所得为与 A、B 同维数数组,其元素 d_ij = b_ij / a_ij
  D=
     ! 1     0.5 !
     ! 0.5   0.25 !
```

例 2.13

```
→ A=[1,2; 2,3];↙
→ b=0.5;        ↙
→ C=A.^B ↙
C=
   !   1       1.4142 !
   ! 1.4142    1.7321 !
```

例 2.14

```
→ A=[1,2; 2,3];↙
→ b=2;          ↙
→ C=b.^A ↙
C=
   ! 2     4 !
   ! 4     8 !
```

例 2.15

```
→ A=[1,2; 2,3];↙
→ b=[ 1,0; 0,1];   ↙
→ C=A.^B ↙
C=
   ! 1     1 !
   ! 1     3 !
```

2.5 与数值型矩阵有关的若干常用函数

2.5.1 常用矩阵的生成函数

例 2. 16

```
→ a = zeros(2,3)
    a =
        ! 0   0   0 !
        ! 0   0   0 !
→ b = ones(2,4)
    b =
        ! 1   1   1   1 !
        ! 1   1   1   1 !
→ c = eye(2,2)
    c =
        ! 1   0 !
        ! 0   1 !
→ d = eye(2,3)
    d =
        ! 1   0   0 !
        ! 0   1   0 !
```

```
→ f = rand(2,2)
    f =
  ! 0.315   0.643 !
  ! 0.276   0.863 !
→ g = rand( )
    g =
        ! 0.728 !
```

2.5.2 size 函数和 matrix 函数

```
y = size(x  [,sel ])
```

```
[nr,nc] = size(x)
```

例 2. 17

```
→ a [1,2,3; 4,5,6]
a =
    ! 1   2   3 !
    ! 4   5   6 !
→ b = size(a)
    b =
        ! 2   3 !
→ b = size(a,1)
    b =
```

```
        ! 2 !
→ b= size(a,2)
    b =
        ! 3 !
→ b= size(a,c)
    b =
        ! 3 !
```

```
y= matrix(x,m,n)
```

例 2.18 将 2×3 的矩阵 a 重新定义为 3×2 的矩阵 b。

```
→ a=[1.5,1.8,2.2;1,3]
    a =
        ! 1.5  1.8  2.2 !
        !  1    2    3 !
→ b= matrix(a,3,2)
    b =
        ! 1.5  2   !
        ! 1    2.2 !
        ! 1.8  3   !
```

2.5.3 从已知矩阵提取部分元素来构成同阶新矩阵的若干函数

```
y= triu(x [,k ])
```

```
y= tril(x [,k ])
```

例 2. 19

```
→ a=[1,2,3; 4,5,6; 7,8,9]
   a=
   ! 1.0  2.0  3.0 !
   ! 4.0  5.0  6.0 !
   ! 7.0  8.0  9.0 !
→ y=triu(a,1)
   y=
   ! 0 2.0  3.0 !
   ! 0 0    6.0 !
   ! 0 0    0   !
→ y=triu(a)
   y=
   ! 1.0  2.0  3.0 !
   ! 0    5.0  6.0 !
   ! 0    0    9.0 !
→ y=triu(a,-1)
   y=
   ! 1.0  2.0  3.0 !
   ! 4.0  5.0  6.0 !
   ! 0    8.0  9.0 !
→ y=tril(a,1)
   y=
   ! 1.0 2.0  0   !
   ! 4.0 5.0  6.0 !
   ! 7.0 8.0  9.0 !
→ y=tril(a,-1)
   y=
   ! 0    0    0 !
   ! 4.0  0    0 !
   ! 7.0  8.0  0 !
```

```
y=diag(x,[,k])
```

例 2. 20

```
→ x=[1,2,3]
  x=
     ! 1.0  2.0  3.0 !
→ y=diag(x)
   y=
   ! 1  0  0 !
   ! 0  2  0 !
   ! 0  0  3 !
→ y=diag(x,1)
   y=
   ! 0  1  0  0 !
   ! 0  0  2  0 !
   ! 0  0  0  3 !
→ x=[1,2,3; 4,5,6; 7,8,9 ]
   x=
```

```
    ! 1  2  3 !
    ! 4  5  6 !
    ! 7  8  9 !
→ y = diag(x,1)
    y =
    ! 2  6 !
→ y = diag(x, - 1)
    y =
    ! 4  8 !
```

2.5.4 与方阵的行列式求值、求逆、线性代数方程组的求解、求矩阵特征值与特征向量等有关的函数

```
① det(x)
② [e,m] = det(x)
```

```
[e,m] = det(x)
```

```
Y = inv(X)
```

```
[x0,KerA] = linsolve(A,b [,x0])
```

$$\begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 3 \\ 5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
→ a = [ 2,1; 1,4]
    a =
        ! 2  1 !
        ! 1  4 !
→ b = [ - 3, - 5]
    b = ! - 3   - 5 !
→ [x0, KerA ] = linsolve(a,b')
      KerA =
          [  ]
      x0 =
          ! 1 !
          ! 1 !
```

$$\begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 3 \\ 6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
→ a = [2  1; 4  2]
→ b = [ - 3   - 6 ]
→ [x0, KerA ] = linsolve(a,b')
      KerA =
          ! - 0. 4472136 !
          !   0. 8944272 !
```

```
x0 =
    ! 1.2 !
    ! 0.6 !
```

$$\begin{bmatrix}2 & 1\\4 & 2\end{bmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}-\begin{pmatrix}3\\6\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}$$

```
→ a=[2  1; 4  2]
→ b=[-3; 0]
→ [x0,KerA]=linsolve(a,b')
  warning: conflicting linear constraints !
  KerA =
       [  ]
  x =
     [  ]
```

```
evals = spec(A)
```

$$\boldsymbol{Ax}=\lambda\boldsymbol{x}$$

```
[Ab [,x [,bs] ] ]=bdiag(A)
```

$$\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3\\2 & 3 & 4\\3 & 4 & 5\end{bmatrix}$$

```
→ A=[1  2  3;  2  3  4;  3  4  5];
→ [Ab, X, bs]=bdiag(A)
  bs =
  ! 1 !
  ! 1 !
  ! 1 !
  x =
  ! 0.3850898   -0.8276709   -0.5082483 !
  ! 0.5595102   -0.1424137    0.8164966 !
  ! 0.7339306    0.5428436   -0.4082483 !
  Ab =
  ! 9.6234754        0.                0. !
  ! 0.         -0.6234754               0 !
  ! 0.                0.      -5.845E-16 !
```

2.5.5　与矩阵（数组）或向量有关的数据统计函数

```
① [n]=max(A)
② [n [, i]]=max(A)
③ [n [, i]]=max(A, 'r')  或  [n[, i]]=max(A, 'c')
④ [n [, i]]=max(A1, A2,…,An)
```

例 2.25

```
→ A=[3,7,9; 1,13,1; 2,4,15];
→ n=max(A)
    n=
      !15!
→[n, i]=max(A)
    i=
      !3  3!
    n=
      !1 5!
→[n, i]=max(A,'r')
    i=
      !1  2  3!
    n=
      !3  13  15!
→ B=[1 2 3; 4 5 6; 7 8 9];
→ n=max(A, B)
    n=
      !3  7  9!
      !4  13  6!
      !7  8  15!
```

① y=sum(x)

② y=sum(x,'r') 或 y=sum(x, 1)

③ y=sum(x,'c') 或 y=sum(x, 2)

例 2.26

```
→ x=[1 2 3; 4 5 6; 7 8 9];
→ y=sum(x)
    y=
    !45!
→ y=sum(x, 1)
    y=
    !12  15  18!
→ y=sum(x, 2)
    y=
    !6!
    !15!
    !24!
```

① y=mean(x)

② y=mean(x,'r') 或 y=mean(x, 1)

③ y=mean(x,'c') 或 y=mean(x, 2)

例 2.27

```
→ x=[1,2,3;4,5,6;7,8,9];
→ y=mean(x)
    y=
    ! 5. !
→ y=mean(x,'r')
    y=
    ! 4. 5. 6. !
→ y=mean(x,'c')
    y=
    ! 2. !
    ! 5. !
    ! 8. !
```

① y=cumsum(x)

② y=cumsum(x,'r') 或 y=cumsum(x, 1)

③ y=cumsum(x,'c') 或 y=cumsum(x, 2)

例 2.28

```
→ a=[1  2  3;  4  5  6;  7  8  9];
→ y=cumsum(a)
y=
    ! 1  14  30 !
    ! 5  19  36 !
    ! 12 27  45 !
→ y=cumsum(a, 1)
y=
    ! 1   2   3  !
    ! 5   7   9  !
    ! 12  15  18 !
→ y=cumsum(a,'c')
y=
    ! 1  3   6  !
    ! 4  9   15 !
    ! 7  15  24 !
```

① y=prod(x)

② y=prod(x,'r') 或 y=prod(x, 1)

③ y=prod(x,'c') 或 y=prod(x, 2)

例 2.29

```
→ a=[1  2  3;  4  5  6;  7  8  9];
→ y=prod(a)
    y=
    !          !
→ y=prod(a,'r')
    y=
    ! 28   80   162 !
→ y=prod(a, 2)
    y=
    !   6 !
    ! 120 !
    ! 502 !
```

2.6 向量与矩阵（数组）元素的引用

例 2.30

```
→ a=[1  2;  3  4;  5  6];
→ a(1, 2)
  a(1, 2)=
  ! 2 !
```

```
→ a(2,1)=12
  a(2,1) =12
→ a
    a=
    ! 1    2 !
    ! 12   4 !
    ! 5    6 !
```

例 2.31

```
→a=[1  2  3;  4  5  6];
→a(6)
    a(6) =
        ! 6.0 !
→a(6)=78;
→a
    a=
      ! 1  2  3 !
      ! 4  5  78!
```

例 2.32

```
→ a=1:5
    a=
    ! 1  2  3  4  5 !
→ a(8)=16
    a=
    ! 1  2  3  4  5  0  0  16 !
```

例 2.33

```
→a=[1  2  3  4  5; 3  4  5  6  7]
    a=
    !1  2  3  4  5!
    !3  4  5  6  7!

→a(1,8)=9
    a=
    !1  2  3  4  5  0  0  9!
    !3  4  5  6  7  0  0  0!
```

例 2.34

```
→a=[3,4,5,6,7,8]
    a=
    !3  4  5  6  7  8!
→a(1)+a(11)
    error!
    invalid    index!
```

例 2.35

```
→x=[1  2  3]
→x([%T  %F  %T])
    x=
    !1  3!
→i=x<5;
→x(i)
    !1  2  3!
```

例 2.36

```
→x=[1  2  3];
    x(S)
    ans=
      !3!
```

```
y=find(条件表达式)
```

例 2.37

```
→A=[1  2  3;  4  5  6];
→rr=find(A>5)
→A(rr)
    ans=
    !6!
```

2. 7　整型数组

例 2. 38

```
→ x=[1  3  5  7  9]
    x=
    ! 1  3  5  7  9 !

→ B=int32(x)
    B=
    ! 1  3  5  7  9 !
```

例 2. 39

```
→ x=int32(9);
→ y=int32(5)
→ z=x/y
    z=
     1
```

例 2. 40

```
→ x=[0  3.2  27  135];
→ x1=int32(x)
    x1=
    ! 0  3  27  135 !
→ x2=int8(x)
    x2=
    ! 0  3  27  -121 !

type(x)
```

2. 8　布尔型向量与矩阵的定义及基本运算

例 2. 41

```
→ a=%T;
→ b=%F;
→ c=%T;
→ d=%F;
→ x=a&b
```

```
    x =
    ! F !
→ x = a & ~ b
    x =
    ! T !
→ x = a | b = = ~ b
    x =
    ! T !
→ b
    b =
    ! F !
→ x = (b = ~ b) | a
    x =
    ! T !
→ b
    b =
    ! T !
```

例 2.42

```
→ a = [ % T, % F, % T, % T, % F];
→ b = [ % F, % F, % T, % F, % T];
→ c = a & b
    c =
    ! F   F   T   F   F !
→ d = a | b
    d =
    ! T   F   T   T   T !
→ f = ~ a
    f =
    ! F   T   F   F   T !
```

例 2.43

```
→ a = 1:5;
→ a == [2   1   3   5   4]
    ans =
    ! F   F   T   F   F !
→ a > 1

  ans =
    ! F   T   T   T   T !
```

```
→ a = [1   2   3.2   4.1   5.6];
→ c = a(a >= 3)
    c =
    ! 3.2   4.1   5.6 !
```

2.9 字符串型数据的定义与运算

2.9.1 字符串的定义

```
--> a = 'abc'
    a =
    ! abc !
```

```
--> b = [ Today'  'I'; "am""very happy"]
    b =
    ! Today    I          !
    !   am     very happy !
```

```
a = code2str(c)
```

```
c = str2code(a)
```

```
string(x)
```

```
n = emptystr(x)
```

例 2.45

```
--> c = str2code('SCILAB')
    c =
    !  - 28 !
    ! 12    !
    ! 18    !
    ! 21    !
    ! 10    !
    ! 11    !
```

```
--> str = code2str(c)
    str =
    ! SCILAB !
--> empty str(str)
    !          !
--> x = [ 3.14159   2.71828]      //实型数据
    x =
    ! 3.14159   2.71828 !
--> x(1) * x(2)
    ans =
    ! 8.5397213 !
--> x = string(x)                 //产生字符型数据
    x =
    ! 3.14159   2.71828 !        // 字符型数组
--> x(1) + x(2)                   // 字符型数据连接运算,“+”为字符串连接运算符
    ans
    3.14159   2.71828
--> x(1) * x(2)                   // 字符型数据不能进行“*”运算
    !  --error    4
    undefined variable : C_ n _ C
```

2.9.2 字符串的运算

例 2.46

```
→ a = string(zeros(2, 2));   b = string(ones(2, 2));
→ c = a + b                                          // 生成字符串矩阵
    c =  ! 01  01 !
         ! 01  01 !
```

```
[txt] = strcat(vstr [, strp])
```

2.9.3 与处理字符串有关的一些常用函数

```
[txt] = strcat(vstr [, strp])
```

例 2.47

```
→ a = string(1:10);
→ b = strcat(a)
    b =
    ! 1  2  3  4  5  6  7  8  9  10 !
→ c = strcat(a, " * ")
    c =
    ! 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10 !
```

```
n = length(str)
```

例 2.48

```
→ a = ["Today" "I am"; "very" "happy"];
→ n = length(a)
    n =
    ! 5  4 !
    ! 4  5 !
```

```
[y] = convstr(str [, flag])
```

例 2.49

```
→ a = ["SCILAB" "is" "a good tool"];
→ b = convstr(a, u)
    b =
    ! SCILAB IS A GOOD TOOL !
```

```
ind = strindex(str1, str2)
```

例 2.50

```
→ a = strindex('How do you do !', 'do')
    a =
    ! 5   12 !
→ b = ['SCI', 'sci']
→ a = strindex('SCI/demos/scicos', b)
    a =
    ! 1   11 !
```

```
a = strsubst(str1, str2, str3)
```

例 2.51

```
→ a = strsubst('2 * % Pi * R * H','R','x')
    a =
    ! 2 * % Pi * x * h !
```

例 2.52

```
→ str = ['2 * %Pi * R * h''%Pi * R * R * H];
→ R = 5;   h = 10;
→ a = evstr(str)
    a =
    ! 314.15927   785.39816 !
```

2.10 多项式类型

2.10.1 多项式的定义

```
→ x = poly(0,"x")          //定义了多项式符号变量 x
→ P = 1 + 2 * x + 3 * x^3  //定义了多项式 1+2x+3x^3,且将它存于 p
```

(2) 直接定义多项式,格式如下:

```
[P] = poly(a,"x"[,"flag"])
```

```
→ a = [1   3];
→ p = poly(a,"x")
    p =
    ! 3 - 4x + x^2 !
```

```
→ x = poly(0,"x")
→ p = 3 - 4 * x + x^2
    p =
    ! 3 - 4x + x^2 !
```

```
-> a=[1  2  1];
-> p=poly(a,"x","coeff")
    p=
    !1+2x+x²!
```

```
-> a=[2  1;  1  3];
-> P=poly(a,"x")
    P=
    !5-5x-x²!
```

```
[p]=inv_coeff(c[,d[,name]])
```

```
d=-1+size(c,'c')/size(c,'r')
```

例 2.56

```
-> a=[1  2  3  4;  5  6  7  8];
-> b=inv_coeff(a)
    b=
    !1+3x  2+4x!
    !5+7x  6+8x!
-> a=[1  3  5  7  9  11;  2  4  6  8  10  12];
-> b=inv_coeff(a)
    b=
    !1+5x+9x²     3+7x+11x²!
    !2+6x+10x²    4+8x+12x²!
-> a=[1  3  5  7  9  11  13  15;  2  4  6  8  10  12  14  16];
-> b=inv_coeff(a,3)
    b=
    !1+5x+9x²+13x³   3+7x+11x²+15x³!
    !2+6x+10x²+14x³  4+8x+12x²+16x³!
```

```
[c]=coeff(p[,d])
```

```
-> c=coeff(p)
    c=
    !1  2  3!
-> c=coeff(p,2)
    c=
    !3!
```

```
->p=1+%s+2*%s^2
    p=

    !1+s+2s²!
```

2.10.2 多项式运算

例 2. 58

```
→ p = poly([1 2],'s')
    p =
    ! 2 - 3s + s² !
→ q = poly([1, 2], 's','c')
    q =
    ! 1 + 2s !
→ p + q
    ans =
    ! 3 - s + s² !
→ q/p
    ans =
    !  1 + 2s  !
    !  ------  !
    !  2 - 3s + s²  !
```

例 2. 59

```
→ x = poly(0,"x")
    x =
    x
→ a = [1 + x   1 + 2 * x + x^2; 3 + 4 * x + x^2   5 + 6 * x];
→ b = [8 + x   2 - x^2; 4 - 5 * x + x^2   1 - x];
→ c = a + b
   c =
   ! 9 + 2x          3 + 2x   !
   ! 7 - x + 2x²     6 + 5x   !
→ c = a * b
   c =
  ! 12 + 12x - 4x² - 3x³ + x⁴        3 + 3x - 2x² - 2x³        !
  ! 44 + 34x - 13x² + 7x³            11 + 9x - 7x³ - 4x³ - x⁴  !
→ l = a * b
   l =
  ! 8 + 9x + x²                  2 + 4x + x² - 2x³ - x⁴ !
  ! 12 + x - 13x² - x³ + x⁴      5 + x - 6x²            !
```

```
→ p = a/b
   p =
     ! 3 + 6x + 2x² - x³      6 + 15x + 11x² + 2x³   !
     ! ------------------     ---------------------  !
     !  - 3x - 4x² + x³         3x + x² - 5x³ + x⁴   !
     !                                               !
     ! 17 + 15x - 7x²       34 + 45x + 7x³ + 4x³ + x⁴ !
     ! ------------------     ---------------------  !
     !  - 3x - 4x² + x³         3x + x² - 5x³ + x⁴   !
→ p = a./b
     p =
     ! 1 + x          1 + 2x + x²   !
     ! -------        ----------    !
     ! 8 + x            2 - x²      !
     !                              !
     ! 3 + 4x + x²      5 + 6x      !
     ! ----------      -------       !
     ! 4 - 5x + x²      1 - x       !
```

2.10.3　有关多项式的几个常用函数

```
[R Q] = pdiv(p1, p2)
```

$$p1_{ij} = Q_{ij} \times p2_{ij} + R_{ij}$$

```
→ x = poly(0,'x')
→ p1 = 2 + 2 * x - 3 * x^2 + 4 * x^3
→ p2  = 6 + 2 * x + x^2
→ [R  Q] = pdiv(p1, p2)
    Q =
     - 11 + 4x
    R =
    68
```

例 2.61

```
→ x = poly(0,'x');
→ p1 = 1 + x + x^2;
→ p2 = 1 + x;
→ q = p1/p2;
→ denom(q)
    ans =
    ! 1 + x !
→ numer(q)
    ans =
    ! 1 + x + x² !
```

a. $p=x^2-3x+2$；

b. $p=x^2+x+1$。

先求解 a：

```
→ x = poly(0,'x')
→ p1 = 2 - 3 * x + x^2;
→ x1 = roots(p1)
  x1 =
  ! 1 !
  ! 2 !
```

```
→ p2 = 1 + x + x^2;
→ x2 = roots(p2)
    x2 =
    !  - 0.5 + 0.8660254i !
    !  - 0.5 - 0.8660254i!
```

```
[Z] = horner(p, x)
```

```
→ x = poly(0,'x');
→ p1 = [1 + x + x^2, x^2 - 1];
→ Z1 = horner(p1, 1)
```

```
    Z1 =
    ! 3   0 !
→ Z2 = horner(2 - 3 * x + x^2, 2)
    Z2 =
    ! 0 !
```

```
→ Z3 = horner(p1, x^2)
    Z3 =
    ! 1 + x^2 + x^4   x^4 - 1 !
```

```
[pd] = derivat(p)
```

```
→ x = poly(0,'x');
→ p = [1 + 5 * x   2 + 3 * x + x^2   3 - 5 * x + x^4]
→ R = derivat(p)
    R =
    ! 5   3 + 2x   5 + 4x³ !
→ p = (x^2 - 1)/(1 + x + x^2);
→ R = derivat(p)
    R =
    !        1 + 4 * x + x^2          !
    ! ------------------------------- !
    ! 1 + 2 * x + 3 * x^2 + 2 * x^3 + x^4 !
```

2.11 表类型

2.11.1 表类型的定义

```
list(a1, a2,…,an)
```

```
→ x = list("xiaoming", 12, "F", "Beijing");     // 定义简单表类型变量 x
→ y = list("200405168", x);     //定义表类型变量 y,其第二个域 x 为前已定义的表变量
→ z = list(abc, 15, [0.8,  0.9])
  ! --error      4
  undlfined    variable:  abc
```

2.11.2 表类型数据的引用，域的插入与删除

```
→ a = x(1)
    a =
    ! xiaoming !
```

```
→ y(2)(1)
    y(2)(1) =
    ! xiaoming !
→ y(2)(4)
    y(2)(4) =
    ! Beijing !
```

```
→ y(2) = list("feifei", 13 "m","Hebei");
```

```
→ y(2) (3) = 'M'
    y(2) (3) =
    ! M !
```

例 2.66 表类型数据的插入。

```
→ x = list(12,"abc")
→ x(1) = 18
    x =
    x(1)
    ! 18 !
    x(2)
    ! abc !
→ x(0) = "Wangming"     //在表的头部插入数据
    x =
    x(1)
    ! Wangming !
    x(2)
    ! 18 !
    x(3)
    ! a b c !
```

```
→ x(1) = null( );
→ x
    x =
```

```
    x(1)
    ! 18 !
    x(2)
    ! a b c !
```

2.11.3 tlist 和 mlist 类型

```
→ x = tlist(['car'  Name'  Number], 'xiali' , 108);
  x =
  x(1)
   ! car Name Number !
  x(2)
   ! xiali !
  x(3)
   ! 108 !
→ x. Name       // 同 x(2)
  ans =
   ! xiali !
→ x. Number
  ans =
  ! 108 !
```

```
→ x=mlist(['car', 'Name', 'Number'], 'xiali', 108);
→ x(2)
    !  -error     4
       undefined  variable: %l_e
→ x('Name')
    ans =
      ! xiali !
→x.name
    ans =
      ! xiali !
```

第 3 章　SCILAB 中的程序设计、脚本文件与函数

3.1　引言

3.2　顺序结构程序设计

3.2.1　赋值语句

```
x=y=5
```

```
x=(a+b+c)/2;
S=sqrt(x*(x-a)*(x-b)*(x-c));
```

```
i=i+1
```

表达式=变量名

3.2.2　输入输出语句

```
[x]=input(message[,"string"])
```

```
→ a=input("Chinese")          //执行此语句,显示屏上显示的是 Chinese
    Chinese → 96              //计算机等待输入中文成绩,96 为用户输入的该学生的
                              //中文成绩,该成绩输入后存储于 a
→ b=input("English")          // 执行此句后,显示屏上将显示的是 English,并出现提示符
    English → 95              // 等待用户输入英语成绩 95 后,将其存于 b
→ c=input("Math")
    Math → 100
→ x=(a+b+c)/3
         x=
          ! 97 !
```

```
--> x = input("What is your name?","string")
    What is your name? --> Huang Duo
```

```
print('file-name',x1 [,x2,…,xn])
```

```
--> x= input("s1")
    s1 --> 3;
```

```
--> y = input("s2")
    s2 --> 4;
--> z = input("s3")
    s3 = 5;
--> s = (s1 + s2 + s3)/ 2;                          // "/"为 SCILAB 中的除法运算符
--> area = sqrt(s * (s - s1) * (s - s2) * (s - s3));    // " * "为 SCILAB 中的乘法运算符
--> print( % io(2),area)
    area =
    ! 6.0 !
```

```
disp(x1 [,x2,…,xn])
```

```
--> a = input("a");
    a --> 1
--> b = input("b");
    b --> 1
--> c = input("c");
    c --> 1
--> r = sqrt(b * b - 4 * a * c);
--> x1 = ( - b+ r)/(2 * a);        //在 SCILAB 中,变量名须写在同一排,不允许出现下标变量
--> x2 = ( - b- r)/(2 * a);
--> disp(x1,x2);
```

3.3 选择结构程序设计

3.3.1 if 语句

```
if 条件表达式
  语句序列
end
```

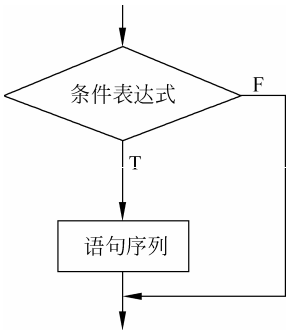

图 3.1 不完整 if 语句的框图

```
→ x= input("length");
→ if x >= 0
    y = x * x;
→ print( % io(2),y)
→end
```

```
if 条件表达式 then 语句序列 end
```

```
if x >= 0  then  y= x * x;  print( % io(2),y)  end
```

```
if x >= 0,  y= x * x;  print( % io(2),y)  end
```

```
if  条件表达式
  语句序列 1
  else
  语句序列 2
  end
```

```
if  条件表达式  then  语句序列 1  else  语句序列 2  end
```

```
if  条件表达式,  语句序列 1  else  语句序列 2  end
```

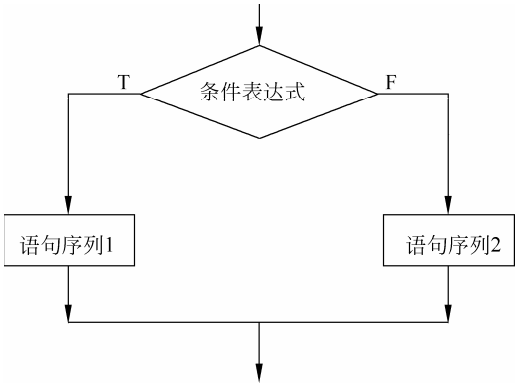

图 3.2　完整 if 语句的框图

$$y=\begin{cases}x^2 & x\geqslant 0\\ x+1 & x<0\end{cases}$$

```
  → x = input('value');
→ if x >= 0
     y = x * x
else
    y = x + 1
end
   print( % io(2),y);
```

3.3.2　select_case 语句

```
select  表达式 0
 case   表达式 1
        指令 1,
 case   表达式 2
        指令 2,
          ⋮
 case   表达式 n
        指令 n,
[else
        指令 n + 1]
end
```

```
select  表达式 0
 case  表达式 1  then  指令 1,
           ⋮
 case  表达式 n  then  指令 n,
      [else      指令 n + 1]
end
```

```
select  表达式 0
 case    表达式 1,  指令 1,
           ⋮
 case    表达式 n,  指令 n,
        [else    指令 n + 1]
end
```

```
n = round(3 * rand( ));
```

```
select  n
case    0    then    disp('0')
case    1    then    disp('1')
case    2    then    disp('2')
else         disp('3')
end
```

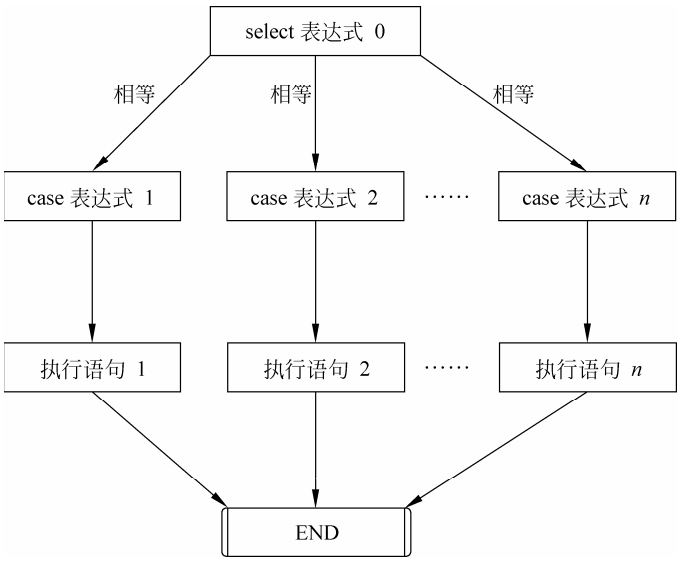

图 3.3　select 语句框图（else 缺省情形）

3.4　循环结构程序设计

3.4.1　for 语句

```
for　循环变量 = 初值:步长:终值
    语句序列(循环体)
end
```

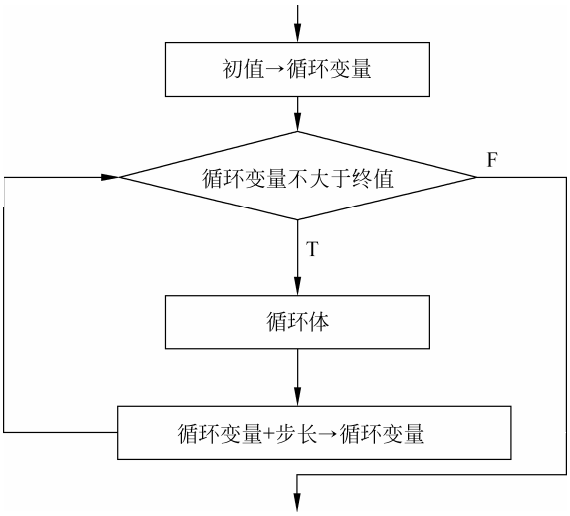

图 3.4　for 语句的框图

```
for　循环变量 = 初值:终值
    循环体
end
```

```
x = 0;
for  i = 1:100
     x = x + i,
end
print(%io(2),x);
```

```
x = 0
for i = [1  2.5  3.2  4.6  7.8]
  x = x + i
end
frint(%io(2),x);
```

3.4.2　while 语句

```
while　条件表达式
    语句序列(循环体)
end
```

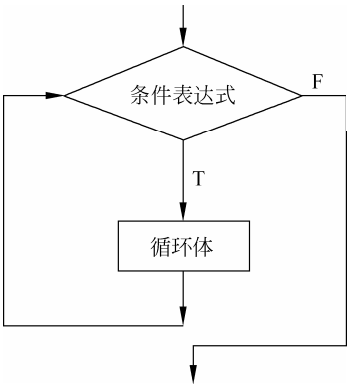

图 3.5　while 型循环框图

```
eps = 1;
while 1 + eps > 1
    eps = eps / 2;
print( % io(2),eps);
    end
```

3.4.3　循环语句的嵌套（多重循环语句）

```
 for i = 1:6
  sum = 0;
  for j = 1:4
   a = input('score');
   sum = sum + a;
   end
  aver = sum/4;
  print( % io(2),sum);
  print( % io(2),aver);
end
```

3.4.4　continue 语句和 break 语句

```
for i = 1:10
 x = i;
  if i>2&i< = 8
   continue
   disp('hello')
  end
  x
end
```

```
for k = 1:3;
     for j = 1:4;
          if  k+j>4  then  break;  else  disp(k);  end
    end
end
```

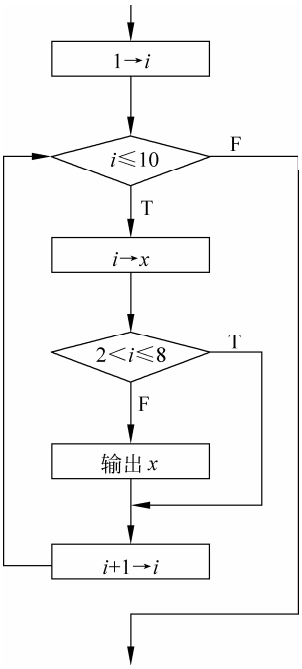

图 3.6　循环体内含有 continue 语句的框图

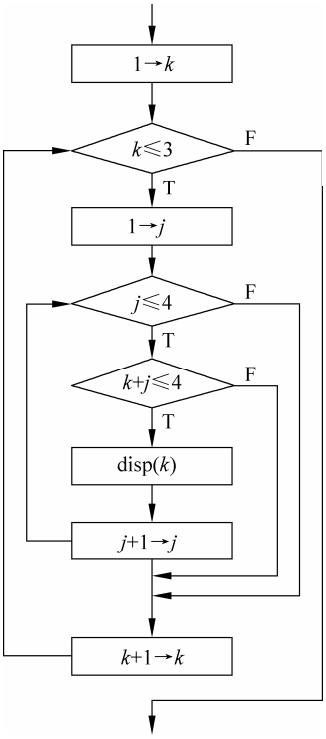

图 3.7　循环体内含有 break 语句的框图

3.5　脚本文件与函数

3.5.1　脚本文件

$$x_{n+1} = \left(x_n + \frac{a}{x_n}\right)\Big/2$$

```
a = input("a = ");   x0 = input("x0 = ");   eps = input("eps = ");
x1 = (x0 + a/x0)/2;
while(abs(x1 - x0)>eps)   //abs(y)为求 y 的绝对值函数
 x0 = x1;
x1 = (x0 + a/x0)/2;
end
disp(x1,"x = ");
```

```
x = 2.236086
```

3.5.2 函数

```
deff('[ s1,s2,…] = function name(e1,e2,…)',text [,opt])
```

```
deff('[x] = myplus(y,z)',  'x = y + z')
```

```
function [y1,y2,…] = function name(x1,x2,…)
            ⋮ }指令或语句序列(函数体)
endfunction
```

```
→ Function  y = area(a,b,c)
→  s = (a + b + c)/ 2;
→  y = sqrt(s * (s - a) * (s - b) * (s - c));
→ endfunction
```

```
function name(s1,s2,…)
```

```
→ area(3,4,5)
ans =
     ! 6 !
 → 2 * 8 + area(3,4,5)
```

```
→ disp(2 * 3.14 *  0.5^2 + area(3,4,5))
```

```
function [y1,y2,… ] = function name(x1,x2,…)
        ⋮ }
          }函数体
        ⋮ }
```

```
function [y1,y2] = root(a,b,c)
    delta = sqrt(b ^ 2 - 4 * a * c);
    y1 = ( - b + delta)/(2 * a);
    y2 = ( - b - delta)/(2 * a);
```

```
[x1,x2] = root(1,5,6)
```

```
getf(filename)
```

```
exec(filename)
```

3.5.3 局部变量与全局变量、函数的嵌套定义与递归调用

例 3.16

```
→ a=5;                          //定义一变量 a 并赋值 5
→ b=3;                          //定义一变量 b 并赋值 3
→ function [y1,y2]=f(x1,x2)
→ b=8;                          //在函数内给变量 b 赋值
→ y1=a+x1+x2;
→ y2=b*x1*x2;
→ a=a+b;                        //在函数内给 a 赋值
→ disp(a);
→ endfunction
→ [p,q]=f(a,b)
   13
   q=
   120
   p=
   13
→ disp(b,'b=',a,'a=')
   a=5
   b=3
```

```
→ global  a;                    //定义变量 a 为全局变量
→ a=3;
→ function y=f(x1,x2)           //定义一函数 f
→ global a;                     // f 内使用了前面定义的全局变量
→ b=x1+x2+a;
→ a=b;
→ y=b;
→ function z=ff(y1,y2)          // 在函数 f 内又定义一函数 ff
→ z=a+y1-y2;                    //此处的 a 为"全局变量",由于没有用 global 说明,故为低层变量
→ a=z;                          //给"全局变量"a——低层变量赋值
→ disp(a,'a=');
→ endfunction                   // 函数 ff 到此结束
→ h=ff(3,4);
→ disp(a,'a=',h,'h=');          //不能将函数 ff 内的 a 的值带回来
→ endfunction                   // 函数 f 到此结束
→ c=f(4,5);
→ disp(a,'a=');                 //由于在函数 f 内用 global 说明了 a,
                                //因此可以将函数 f 内给 a 赋的值带回来
```

```
n=1;
for  i=1:n
  n=n+i;
end
```

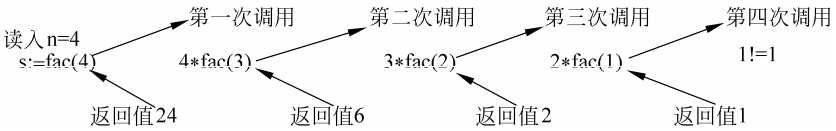

图 3.8　用递归调用求　n!示意图（n　=4）

```
function y = fac(n)
    if n == 1 then y = 1
            else y = n * fac(n - 1)
    end
endfunction
```

3.5.4　有关程序调试的几个常用指令

```
pause
```

```
[x1,x2,…,xn] = resume(a1,a2,…,an)
```

```
→ a = 1;   b = 2;   c = 3;
→ function       y = f(x1,x2)
→ a1 = 2;   a2 = 4;   a3 = 6;
→ y = a1^2 + a2^2 + a3^3 + x1 * x2;
→ [a,b,c] = resume(a1,a2,a3);          // 将函数内数据传回到函数外，
                                       //达到高层数据低层引用的目的
→ endfunction
→ z = f(a,b)
   z =
   58
→ disp(a,b,c)
     6
     4
     2
```

```
→ a = 2;   b = 4;   c = 6;
→ function       y1 = ff(x1,x2)
→ aa = 2;   bb = 4;   cc = 6;
→ y1 = aa^2 + bb^2 + cc^2 + x1 * x2;
→ disp(y1);
→ pause;
→ y1 = aa^2 + bb^2 + cc^2 + x1 * x2;
→ endfunction
→ z = ff(a,b)
   64
-1→ [aa,bb,cc] = resume(a + b,a * c,c) //改变函数内局部变量的值
   z =
   224
```

3.6　函数的应用

3.6.1　函数名作形式参数——二分法求非线性方程的根

```
function B= f(a,b,eps,p)        // p 为方程 p(x)－0 左边的函数，
                                // 此处函数名 p 作为函数 f 的形式参数
A = p(a);  B = A;
while(abs(b - a)> eps | abs(B)> eps)
   x = (a + b)/2;
   B = p(x);                    // 调用函数 p,计算其在 x 处的值
   If(B == 0),  break;
   else if(A * B> 0)  a = x;
        else  b = x
        end
   end
end
```

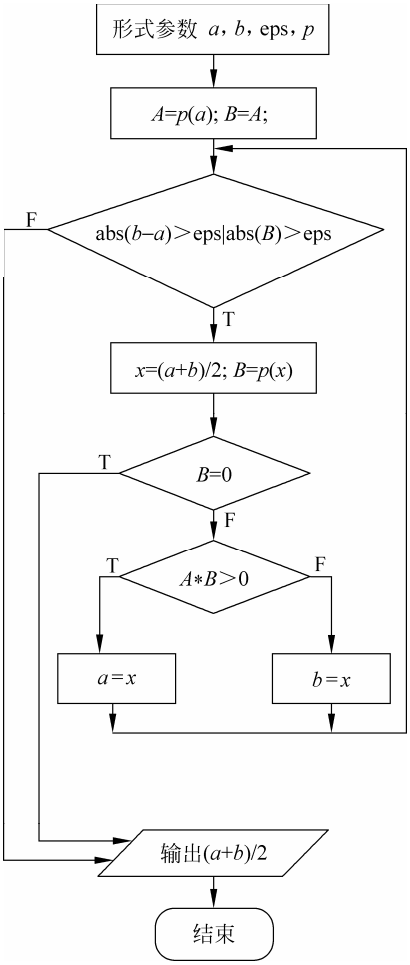

图 3.9　用二分法求方程 p（x）=0 的根的框图

```
function  z = f1(x)
   z = x^3 - x - 1;
endfunction
```

```
function  B = f(a,b,eps,p)        // p 为方程 p(x)=0 左边的函数，
                                  // 此处函数名 p 作为函数 f 的形式参数
A = p(a);     B = A;
while(abs(b - a) > eps | abs(B) > eps)
   x = (a + b)/2;
   B = p(x);                      // 调用函数 p，计算其在 x 处的值
   If(B== 0),  break;
   else  if(A * B> 0)  a = x;
         else  b = x;
         end
  end
end
disp((a + b)/2,'root = ');
endfunction

function  z = f1(x)
   z = x^3 - x - 1;
endfunction
```

```
exec done
```

```
→ a = 1; b = 2;
→ r = 0.000001;
→ f(a,b,r,f1);

root = 1.3247166
```

3.6.2　函数的递归调用——求两正整数的最大公因子

$$m = nq_0 + r_0$$

$$n = r_0 q_1 + r_1$$

```
m = input("m = ",m);
n = inPut("n = ",n);
if(m<n)then a = m; m = n; n = a; end     //交换 m，n 的值，使 m 为 m，n 中的最大者
m = int32(m); n = int32(n);              //使 m，n 为整型数据
r = gcd(m,n);
disp("greatest common factor",r);

function gcdn = gcd(x,y)
    if y = 0  then  gcdn = x;
    else  gcdn = gcd(y,x - x/y * y);     //注意到 x，y 为整型，x/y 为整除，x-x/y * y 为求余
    end
endfunction
```

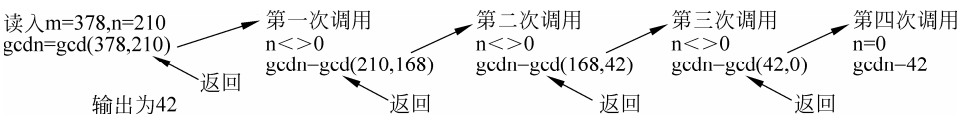

图 3.10　用递归调用求 n!示意图

3.6.3　多重循环的应用——线性方程组的顺序消元法

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{3.1}$$

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & a_{23}^{(0)} & \cdots & a_{2n}^{(0)} & b_2^{(0)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & a_{n3}^{(0)} & \cdots & a_{nn}^{(0)} & b_n^{(0)} \end{bmatrix} \tag{3.2}$$

$$a_{ij}^{(1)} = a_{ij}^{(0)} - \frac{a_{i1}^{(0)}}{a_{11}^{(0)}} \times a_{1j}^{(0)} \qquad (j = 2,3,\cdots,n)$$

$$b_i^{(1)} = b_i^{(0)} - \frac{a_{i1}^{(0)}}{a_{11}^{(0)}} \times b_i^{(0)}$$

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix} \tag{3.3}$$

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{i2}^{(1)}}{a_{22}^{(1)}} \times a_{2j}^{(1)} \qquad (j = 3,4,\cdots,n)$$

$$b_i^{(2)} = b_i^{(1)} - \frac{a_{i2}^{(1)}}{a_{22}^{(0)}} \times b_i^{(1)}$$

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & & & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ 0 & a_{22}^{(1)} & & & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ & & \ddots & & & & \\ \vdots & & & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} & b_k^{(k-1)} \\ 0 & \cdots & & a_{(k+1)k}^{(k-1)} & \cdots & a_{(k+1)n}^{(k-1)} & b_{k+1}^{(k-1)} \\ & & & & \ddots & & \\ 0 & \cdots & & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} & b_n^{(k-1)} \end{bmatrix} \tag{3.4}$$

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \times a_{kj}^{(k-1)} \qquad (j = k+1,\cdots,n) \\ b_i^{(k)} &= b_i^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \times b_k^{(k-1)} \end{aligned} \tag{3.5}$$

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ 0 & a_{22}^{(1)} & & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ & & \ddots & & & \\ \vdots & & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} & b_k^{(k-1)} \\ & & & \ddots & & \\ 0 & & \cdots & & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{bmatrix} \tag{3.6}$$

$$\begin{cases} a_{11}^{(0)}x_1 + a_{12}^{(0)}x_2 + \cdots + a_{1n}^{(0)}x_n = b_1^{(0)} \\ a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \cdots \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases} \tag{3.7}$$

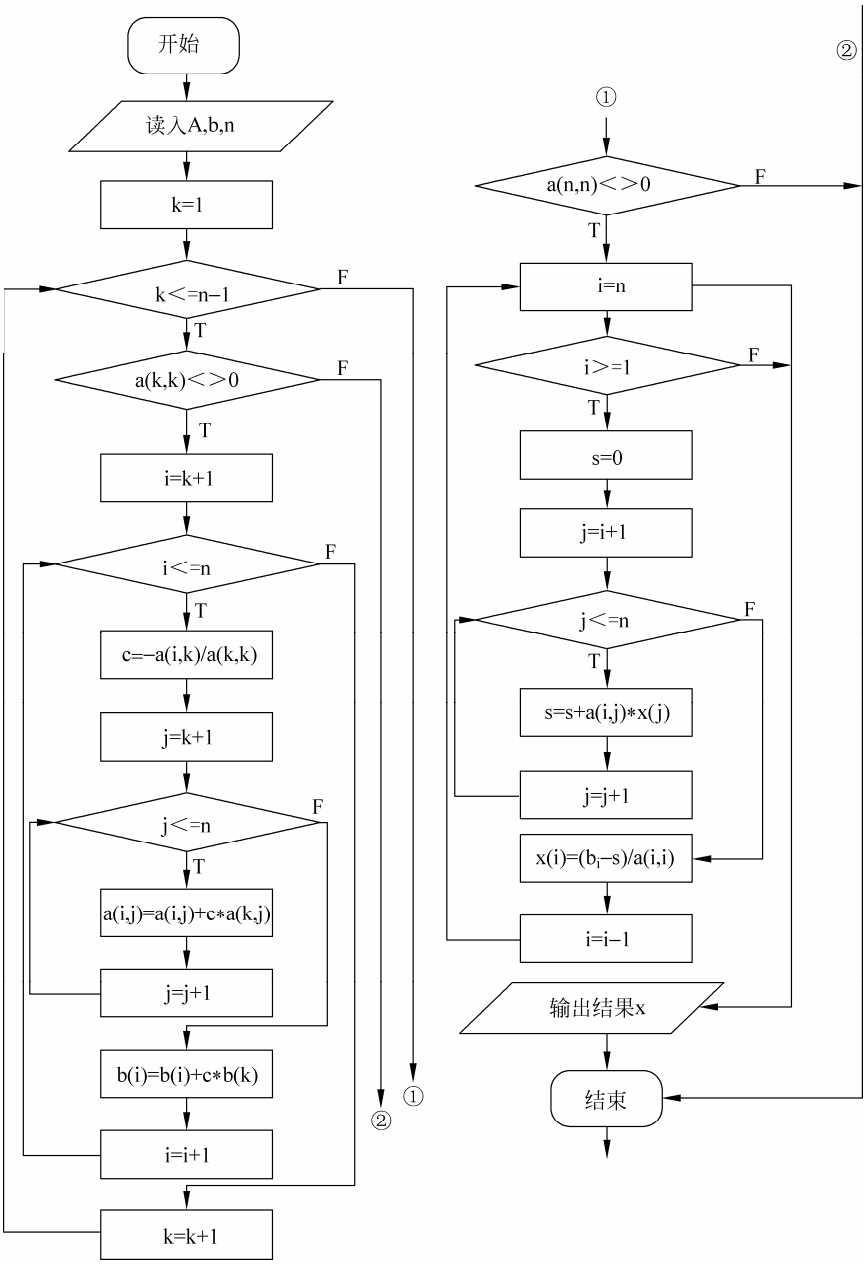

图 3.11　顺序消元法的框图

```
for k = 1:n - 1
    for i = k + 1:n
            c = - a(i,k)/a(k,k);
        for j = k + 1:n
             a(i,j) = a(i,j) + c * a(k,j);
        end
        b(i) = b(i) + c * b(k);
    end
end
```

```
for i = n:-1:1
    sum = 0;
    for j = i+1:n
        sum = sum + a(i,j) * x(j);
    end
    x(i) = (b(i) - sum)/ a(i,i);
end
```

```
function [x] = GD(a,b,n)
  for k = 1:n-1
    for i = k+1:n
      c = -a(i,k)/a(k,k);
      for j = k+1:n
        a(i,j) = a(i,j) + a(k,j) * c;
      end
      b(i) = b(i) + c * b(k);
    end
  end
  for i = n:-1:1
    sum = 0;
    for j = i+1:n
      sum = sum + a(i,j) * x(j);
    end
    x(i) = (b(i) - sum)/ a(i,i);
  end
endfunction
```

第 4 章　计算结果可视化

4.1　引言

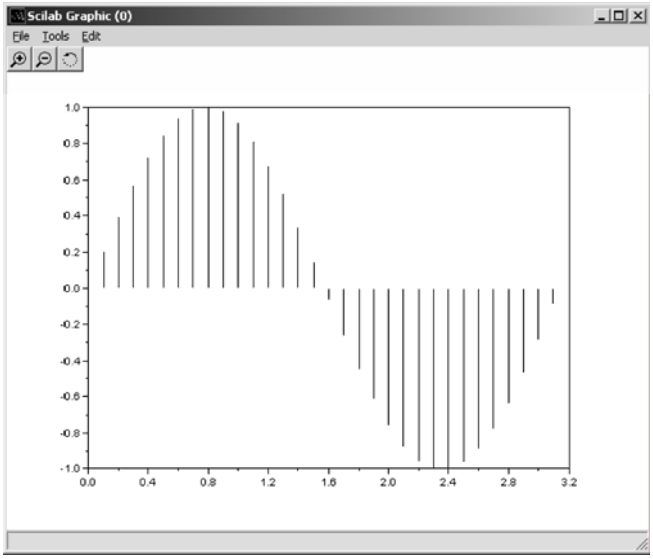

图 4.1　图形窗口

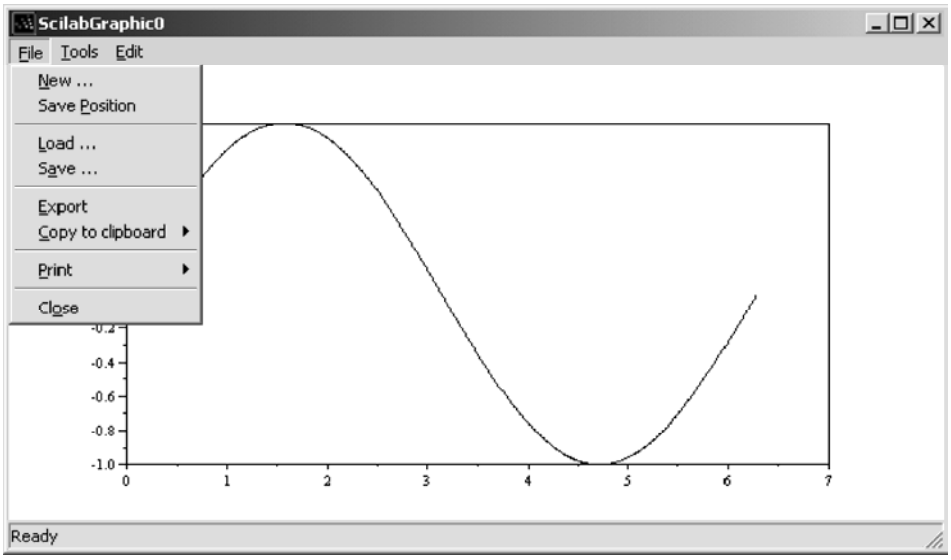

图 4.2　图形窗口的文件菜单

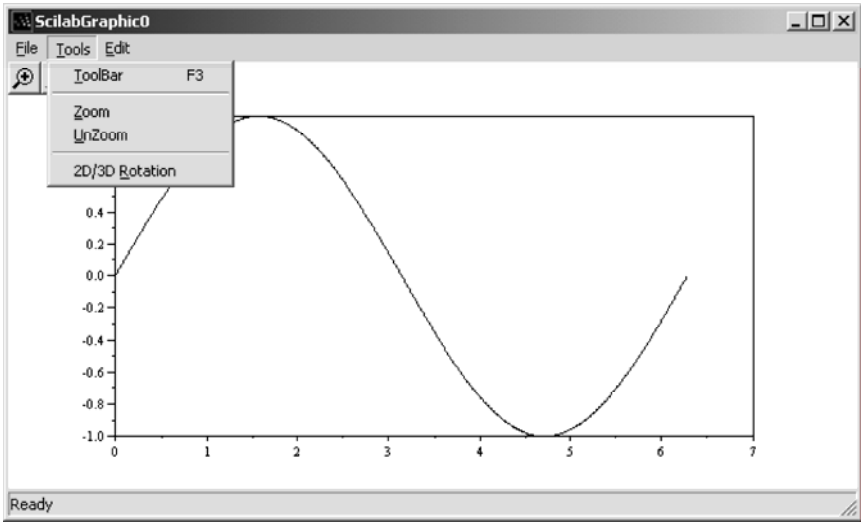

图 4.3　图形窗口的工具菜单

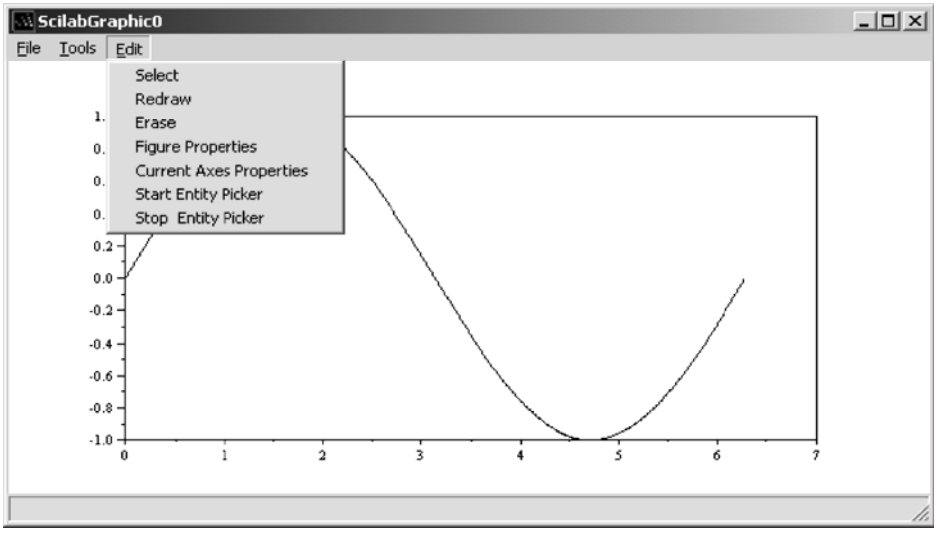

图 4.4　图形窗口的编辑菜单

4.2　二维图形的绘制

4.2.1　plot 指令

```
(1) plot(y [,xcap,  ycap,  caption])
(2) plot(x,y [,xcap,  ycap,  caption])
```

例 4.1

```
→ y = [0.0:0.1:5];
→ plot(y,'X','Y','y = f(x)');
```

例 4.2

```
→ x = 0.0:0.01: %Pi;
→ x1 = %Pi:0.01:2 * %Pi;
→ plot([2 * x - 5; - 5 * x1 + 8]);
```

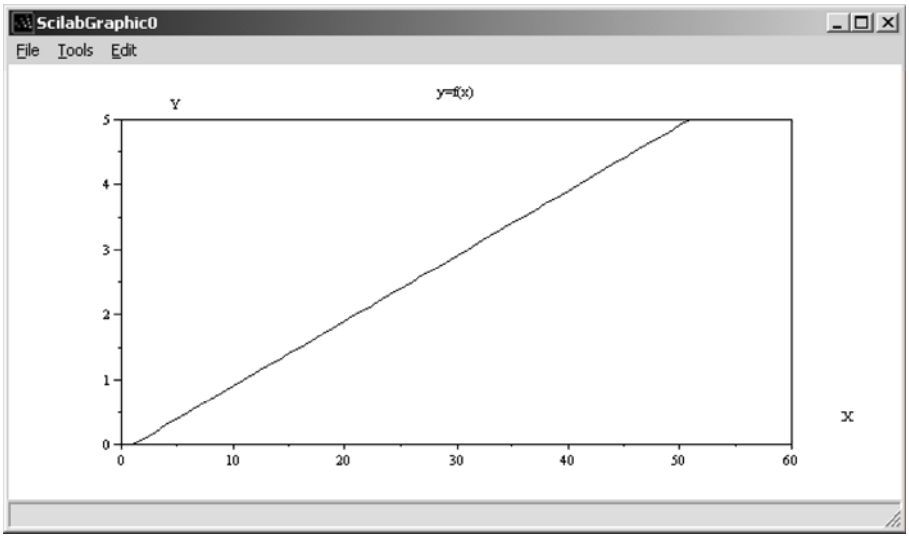

图 4.5　利用指令 plot（y,'X', 'Y', 'y=f（x）'）画图

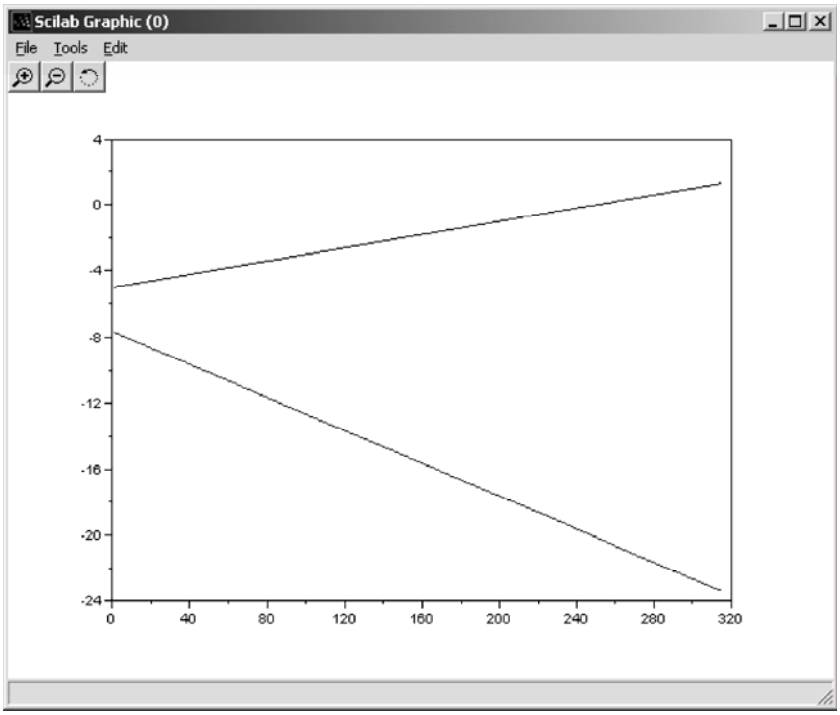

图 4.6　利用指令 plot 在同一窗口画两条曲线

```
plot(x,y [,xcap, ycap, caption])

plot(x1,y1 [,xcap, ycap, caption])

plot(x2,y2 [,xcap, ycap, caption])
...
plot(xn,yn [,xcap, ycap, caption])
```

```
→ x= 0.0:0.01: %Pi;
→ plot(x,sin(x));
```

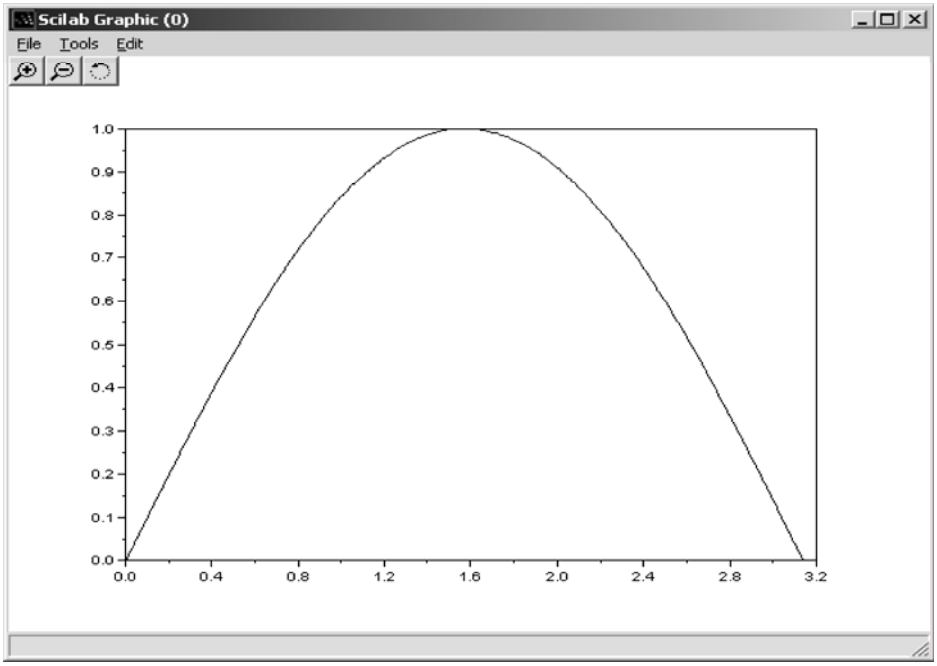

图 4.7　利用指令 plot（x,y）画图

```
→ x = 0.0:0.01: %pi;
→ x1 = %pi:0.01:2 * %pi;
→ z = [x; x1]';
→ y = [sin(x); cos(x1)]';
→ plot(z,y);
```

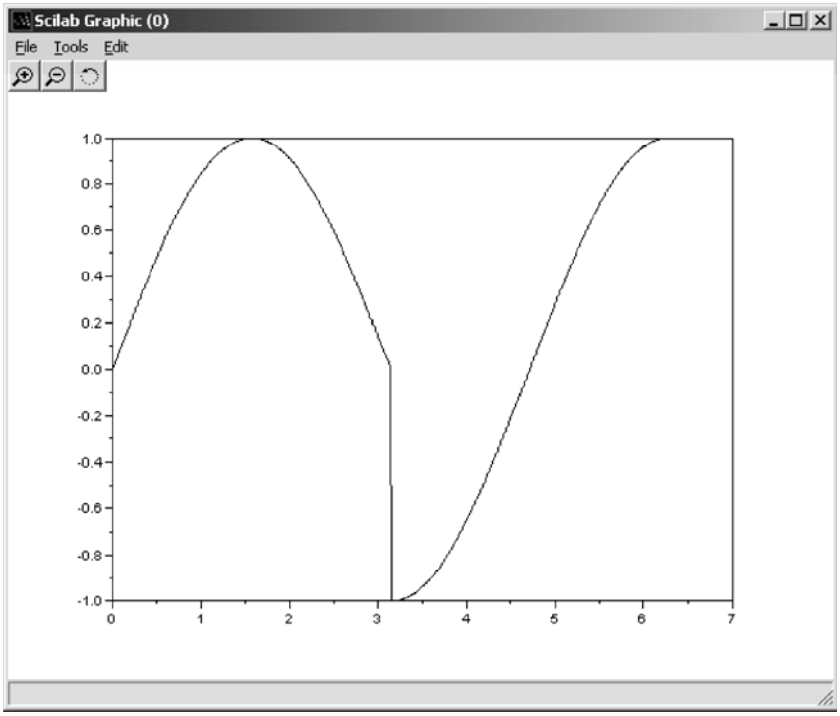

图 4.8　利用指令 plot（z,y）在同一窗口画两条曲线

4.2.2　plot2d 指令

```
→ x = 0.0:0.01: %pi;
→ plot2d(x,sin(x));
```

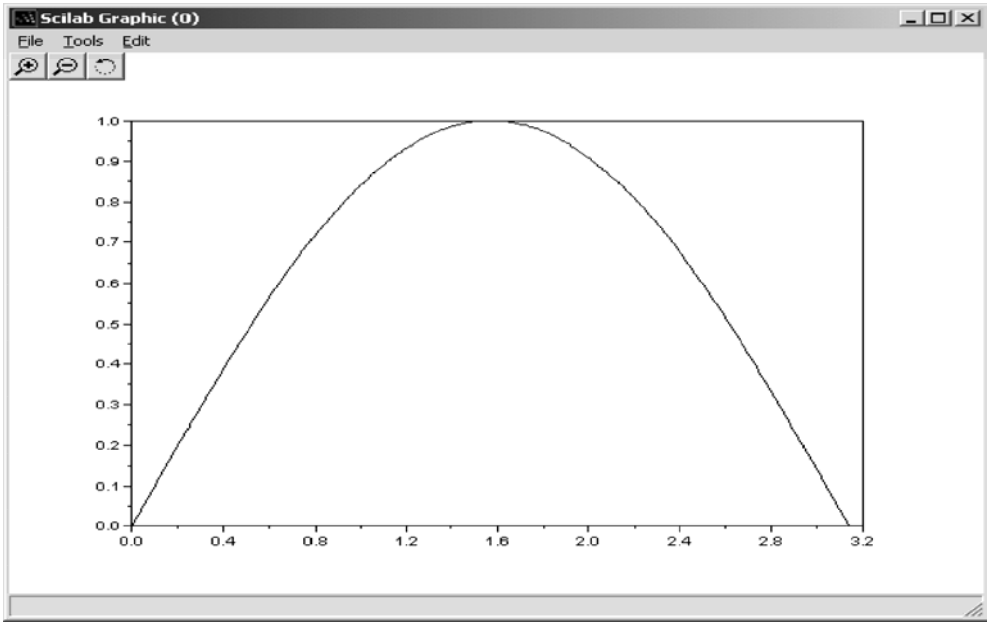

图 4.9　利用指令 plot2d（x,y）画 y=sinx 的图形

```
→ x=0.0:0.01:%pi;
→ x1=%pi:0.01:2*%pi;
→ plot2d([x:x1],[sin(x):cos(x1)]);
```

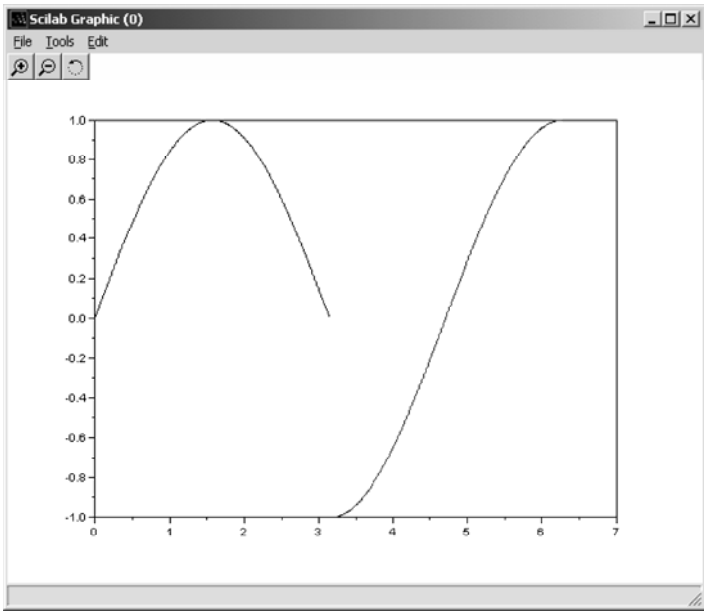

图 4.10　利用指令 plot2d（x,y）在同一窗口画两条曲线

```
→ x=0.0:0.01:%pi;
→ plot2d(x,sin(x),  rect=[0,0,%Pi,1],  leg='sinx',  axesflag=1);
```

```
→ x=0.0:0.01:%pi;
→ plot2d(x',[sin(x):sin(2*x):sin(3*x)]',…,
style=[4,5,6],  rect=[0,-2,2*%Pi,2],  leg="sinx@sin2x@sin3x",axesflag=1);
```

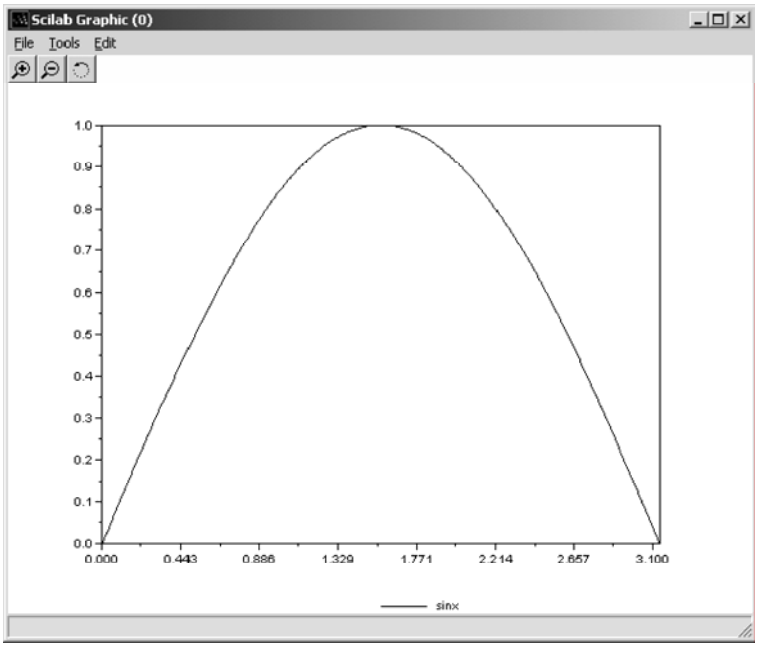

图 4.11　利用指令 plot2d（x,y,<opt_args>画曲线 y=sinx

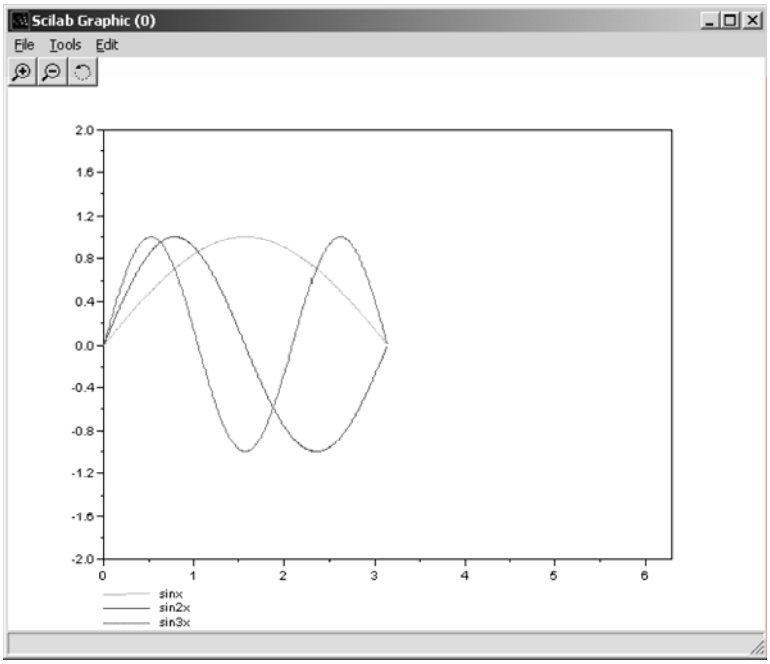

图 4.12　利用指令 plot2d（x,y,<opt+args>）在同一窗口画三条曲线

```
→ x=0.0:0.1:%Pi;
→ plot2d2(x,sin(2*x));
```

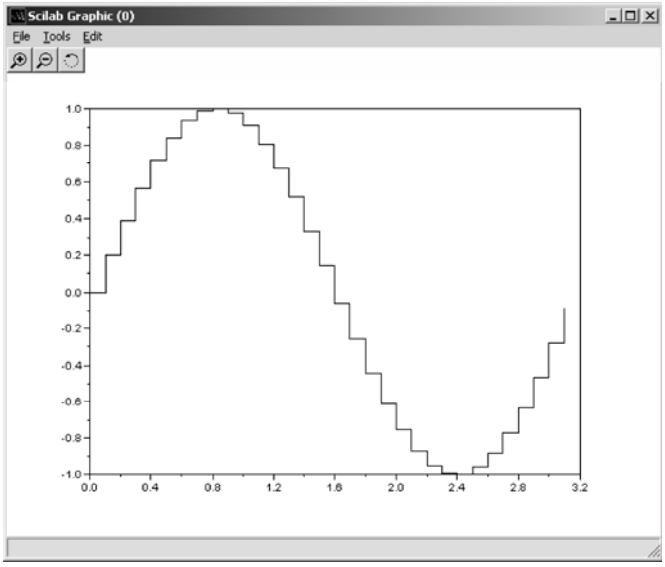

图 4.13　利用指令 plot2d2（x,y）画曲线 y=sin2x

```
→ x=0.0:0.1:%Pi;
→ plot2d3(x,sin(2*x));
```

```
→ x=0.0:0.1:%Pi;
→ plot2d4(x,sin(2*x));
```

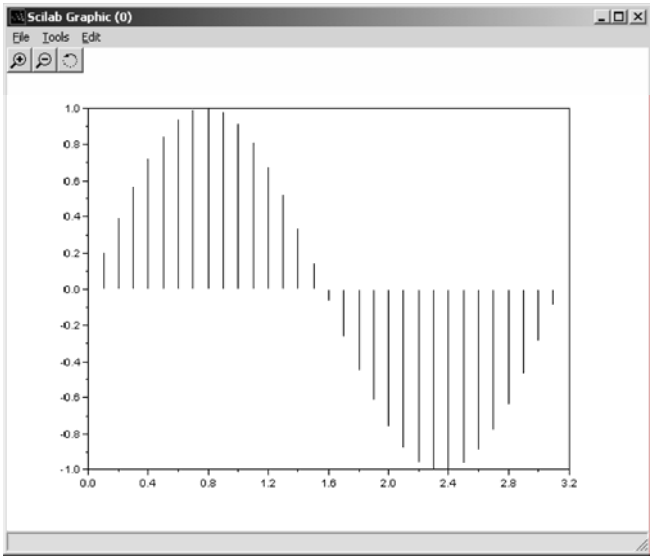

图 4.14　利用指令 plot2d3（x,y）画曲线 y=sin2x

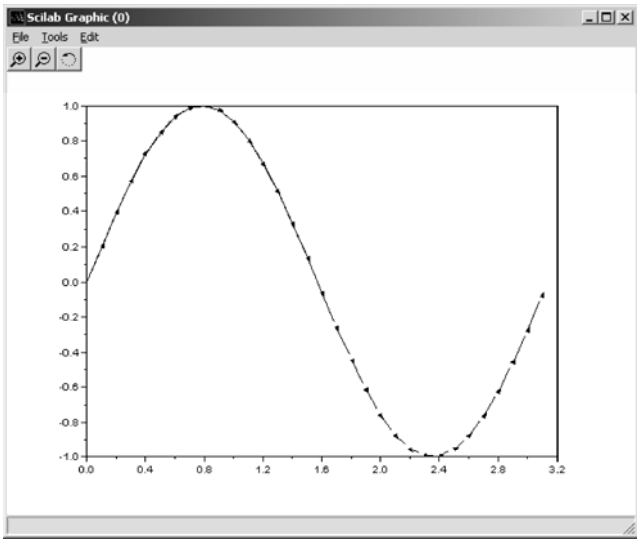

图 4.15　利用指令 plot2d4（x,y）画曲线 y=sin2x

4.3　三维图形的绘制

4.3.1　函数 plot3d（　）——三维曲面的绘制

```
    plot3d(x,y,z [,theta,alpha,leg,flag,ebox])

→ x = 0.0:0.1:2 * %pi
→ y = x;
→ z = sin(x)' * cos(y);
→ plot3d(x,y,z,35,45,"X@Y@Z",[2,2,4]);
```

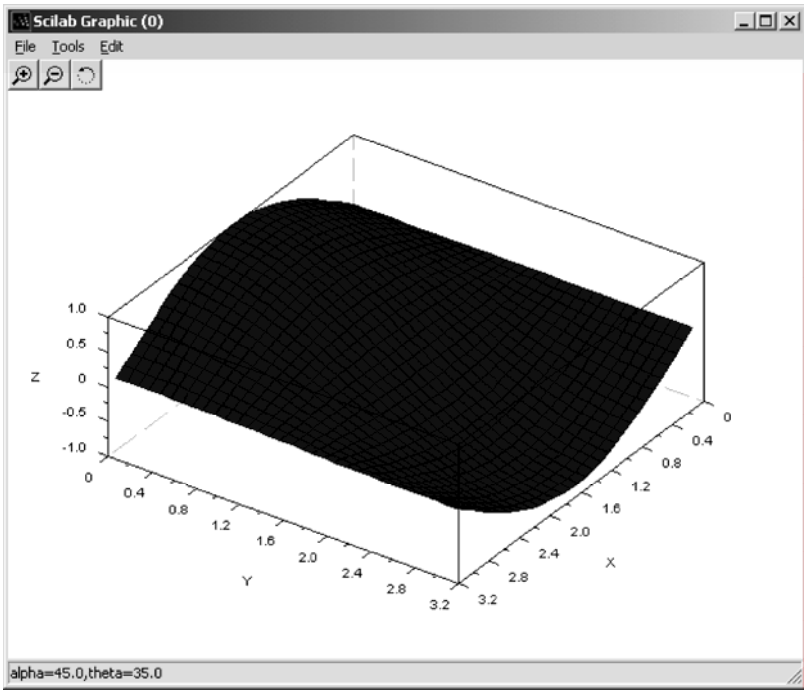

图 4.16　利用指令 plot3d（x,y…）画曲面 z=sinxcosy

```
→x = -10:0.1:10;
→ y = x;
→ for i = 1:201
→ for j = 1:201
→ zz(i,j) = x(i) * x(i) + y(j) * y(j);
→ end
→ end
→ plot3d(x,y,zz,35,45,"X@Y@Z",[2,2,4]);
```

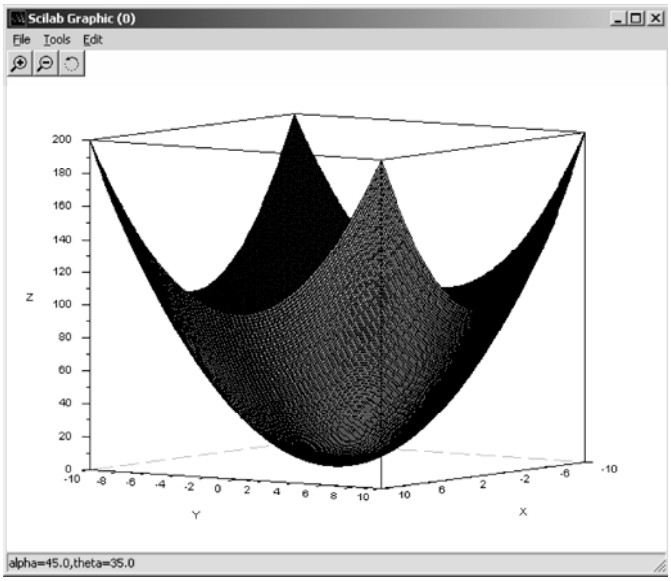

图 4.17　利用指令 plot3d（x,y…）画曲面 z=x2+y2

```
-> x = -10:0.1:10;
-> y = x;
-> for i = 1:201
-> for j = 1:201
-> z(i,j) = x(i) * y(j);
-> end
-> end
-> plot3d(x,y,z,35,45,"X@Y@Z",[2,2,4]);
```

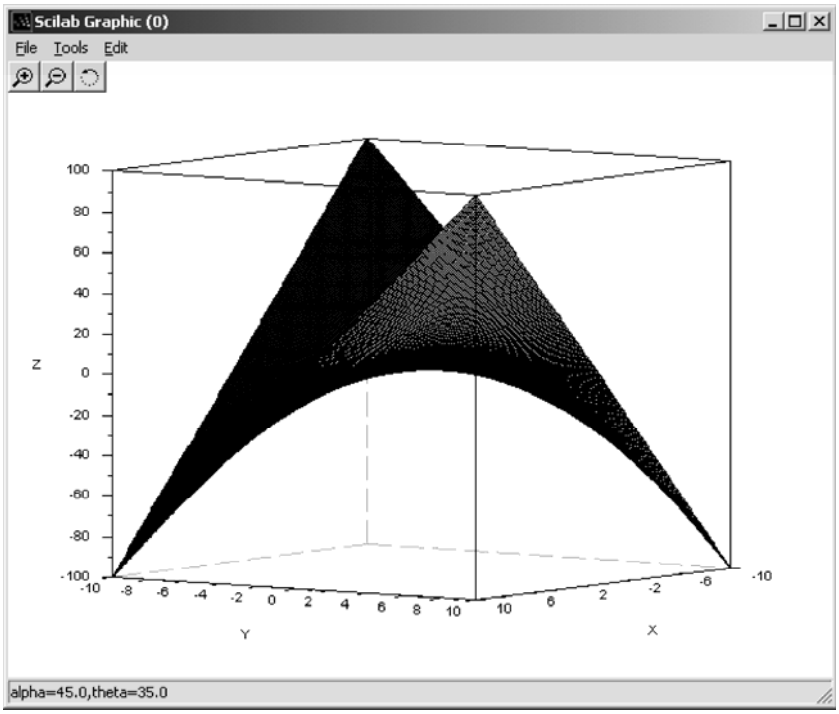

图 4.18　利用指令 plot3d（x,y…）画曲面 z=xy

```
plot3d(xf,yf,zf [,theta,alpha,leg,flag,ebox])

plot3d(xf,yf,zf,<opt_args>)

[xx,yy,zz] = genfac3d(x,y,z [,mask])
```

```
-> x = [0.0:0.1:2 * %pi]; z = sin(x)' * cos(x);
-> [xx,yy,zz] = genfac3d(x,x,z);
-> plot3d(xx,yy,zz);
```

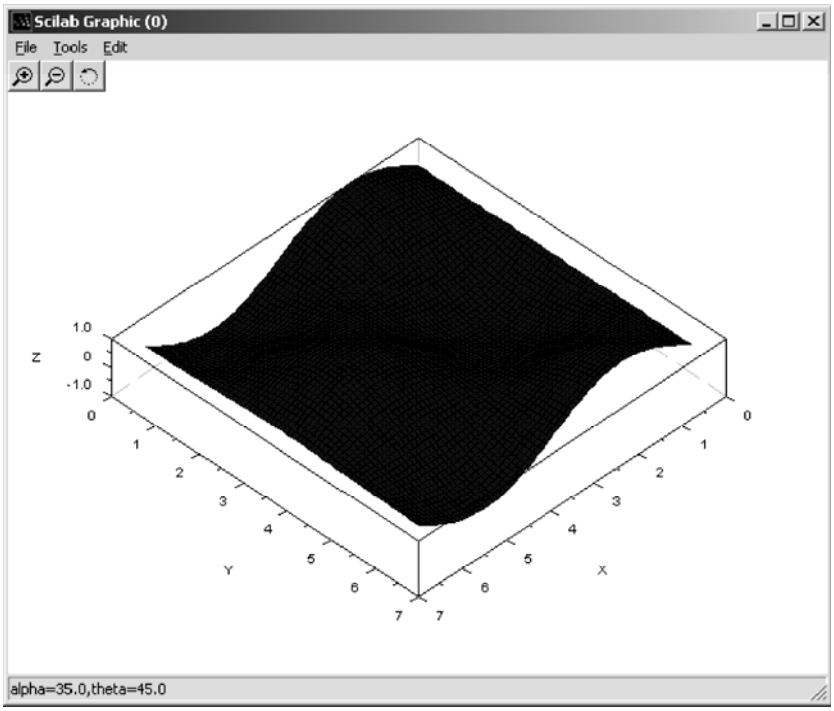

图 4.19　利用指令 plot3d（x,y…）画曲面 z=sinxcosy

```
plot3d(xf,yf,list(zf,colors) [,theta,alpha,leg,flag,ebox])
```

```
→ x = 0.0:0.1:2 * %pi;
→ y = x;
→ z = sin(x)' * cos(y);
→ [xx,yy,zz] = genfac3d(x,y,z);
→ plot3d([xx  xx],  [yy  yy],  list([zz  zz + 6],…,
  [4 * ones(1,3844),  5 * ones(1,3844)]));
```

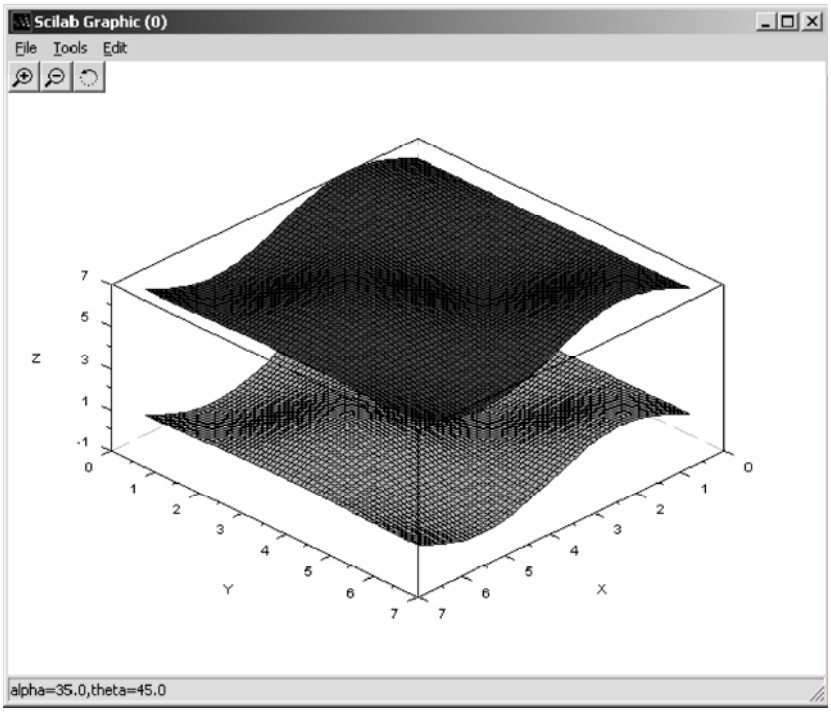

图 4.20　利用指令 plot3d 在同一坐标系中画两张曲面

4.3.2　函数 param3d——三维空间曲线的绘制

```
param3d(x,y,z [,theta,alpha,leg,flag,ebox])
```

```
-->x = 0.0:0.1:5 * %pi;
-->param3d(sin(x),cos(x),x * 0.1,35,45,"X@Y@Z",[2,3]);
```

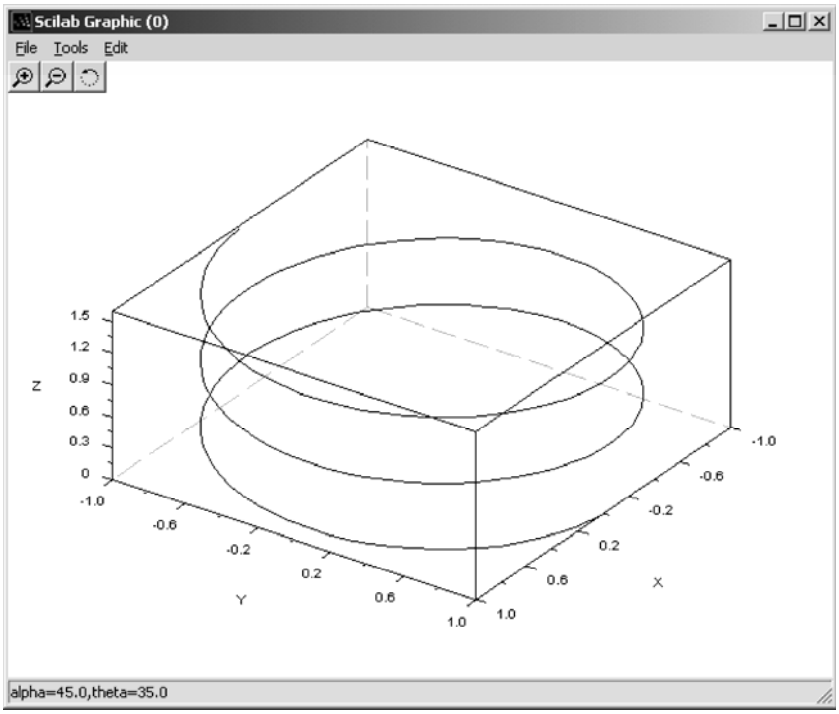

图 4.21　利用指令 param3d 画螺旋线图

```
param3dl(x,y,z [,theta,alpha,leg,flag,ebox])

param3dl(x,y,list(z,colors) [,theta,alpha,leg,flag,ebox])

x = 0.0:0.1:5 * %pi;
param3dl([sin(x); sin(2 * x)]',[cos(x); cos(2 * x)]',list([0.1 * x; sin(x)]',...
[3,2]),35,45,"X@Y@Z",[2,3]);
```

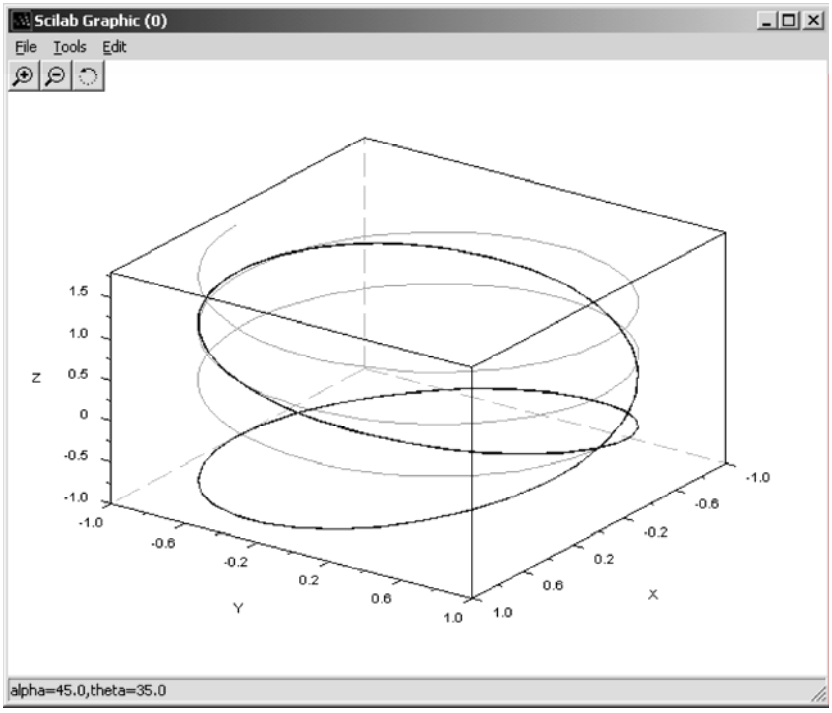

图 4.22　利用指令 param3d 在同一坐标系内画多条空间曲线

4.4　绘图全局参数与色图的设定

4.4.1　绘图全局参数的设定

```
xset([choice-name,x1,x2,x3,x4,x5])
```

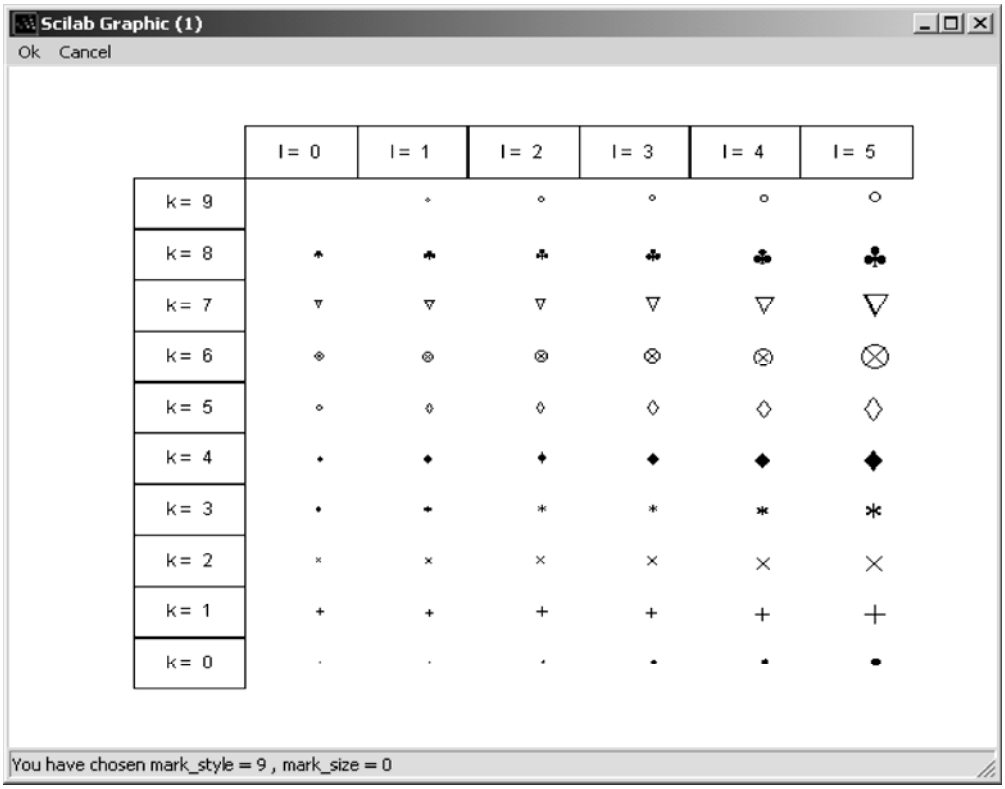

图 4.23　绘图标识类型

```
→ xset("linestyle",1)
→ xset("thickness",1)
→ x=[-2:0.1:2];
→ plot2d(x,x.^3);
→ xbasc( );
→ xset("linestyle",2);

→ plot2d(x,x.^3);
```

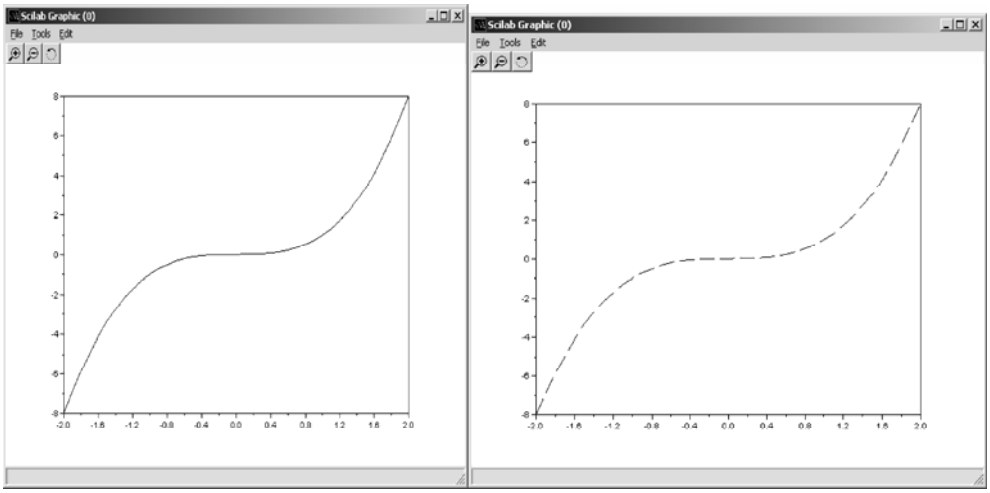

图 4.24　利用不同线型绘制的 y=x3

```
xset("mark-size",5);
x = -2:0.1:2;
y = x.^2;
z = [-0.8+y; -0.6+y; -0.4+y; -0.2+y; y; 0.2+y; 0.4+y; 0.6+y; 0.8+y]';
plot2d(x,z, style=[-9,-8,-7,-6,-5,-4,-3,-2,-1]);
```

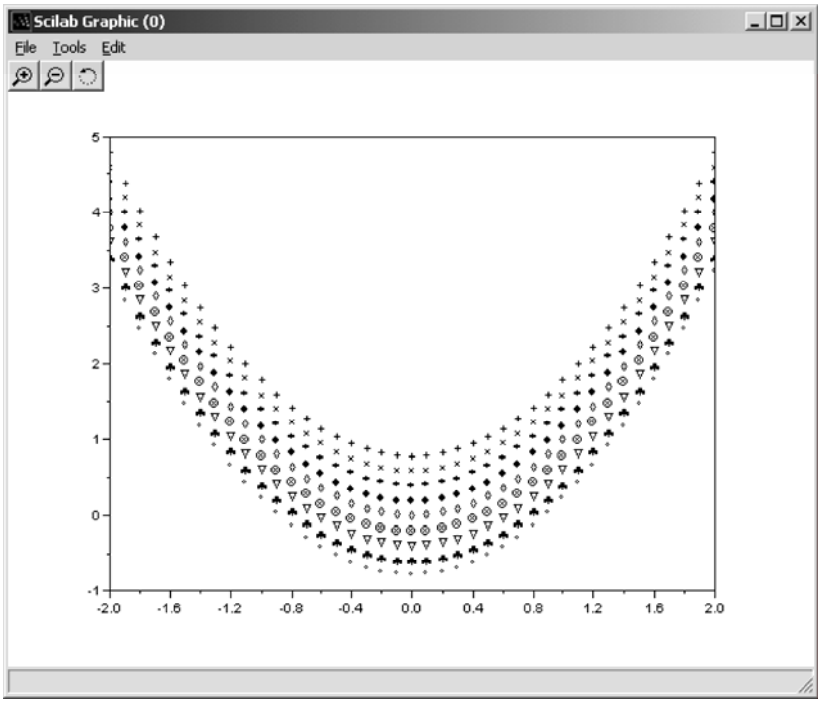

图 4.25　利用不同标识绘制 y=x2+k

4.4.2　色图的设定

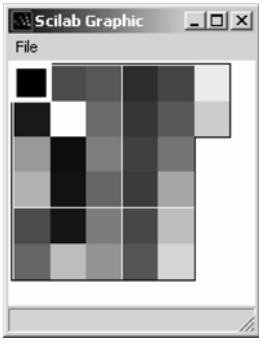

图 4.26　默认时色图的设定

表 4.1　当前色图所对应各颜色的名称

色图号	颜色名称	色图号	颜色名称	色图号	颜色名称
0	黑色	11	深蓝色	22	深紫色
1	黑色	12	天空蓝	23	鲜紫色
2	深蓝色	13	深绿色	24	深棕红色
3	淡绿色	14	深绿色	25	深棕红色
4	天空蓝	15	鲜绿色	26	棕红色
5	鲜红色	16	深绿色	27	深橙色
6	紫色	17	深蓝绿色	28	粉红色
7	鲜红色	18	蓝绿色	29	粉红色
8	白色	19	深红色	30	粉红色
9	淡蓝色	20	深红色	31	粉红色
10	蓝色	21	红色	32	深黄色

```
-->n=228;
-->n=fix(3/8 * n);
-->r=[(1:n)'/n; ones(m-n,1)];
-->g=[zeros(2*n,1); (1:n)'/n; ones(m-2*n,1)];
-->b=[zeros(2*n,1); (1:m-2 * n)'/(m-2 * n)];
-->h=[r,g,b];
-->xset("colormap",h);
-->plot3d1( );
```

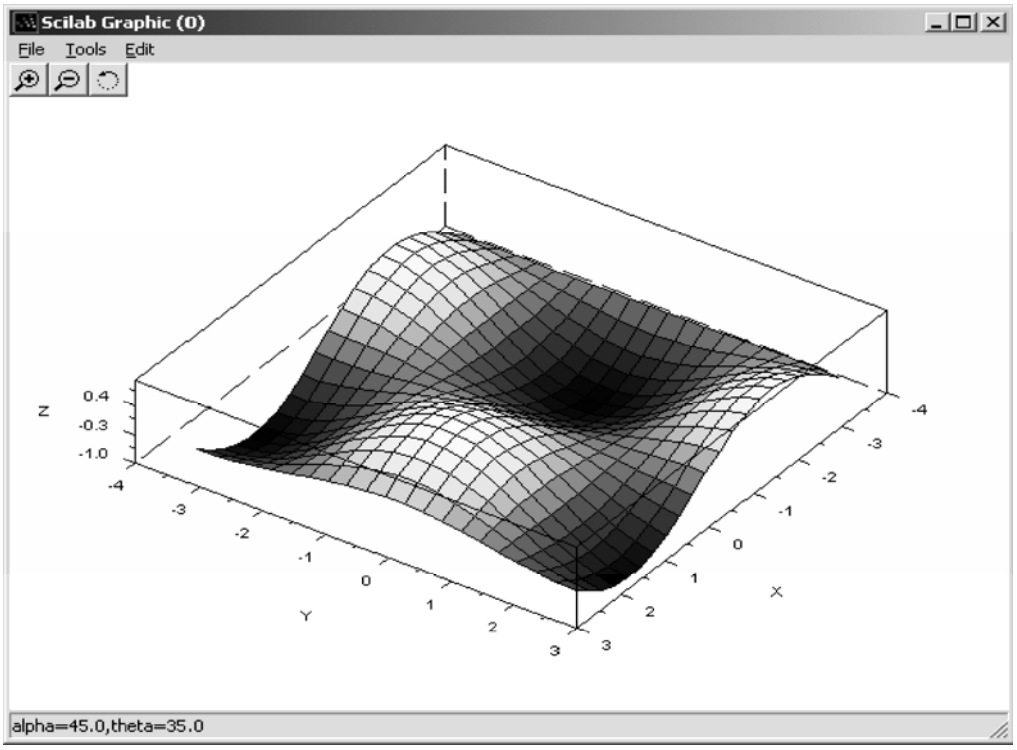

图 4.27　通过设置色图用 plot3d1 画出的图

第 5 章　SCILAB 与 C 或 FORTRAN 程序的接口

5.1　引言

5.2　应用动态链接指令 link

```
(1) link(files,sub-name)
(2) link(files,sub-name,flag)
(3) link(x,sub-names [,flag])
```

```
# define  A(i,k)  a[i*m+k]
# define  B(k,j)  b[k*l+j]
# define  C(i,j)  c[i*l+j]
void  matmul(a,n,m,b,l,c)
    double a[],  b[],  c[];
    int  n,m,l;
    {
      int i,j,k;
      double s;
      for(i=0; i< n; i++)
      {
        for(j=0; j < l; j++)
        {
          s=0;
          for(k=0; k < m; k++)
            { s+ =A(i,k) * B(k,j);
            }
            c(i,j)=s;
        }
    }
}
```

```
→ link("C:\c-program\ matmul.obj","matmul",'c')
```

5.3　调用动态链接程序的指令 call

```
[yl,…,yk]=call("ident",xl,Pxl,"txl",…,xn,Pxn,"txn","out",[nyl,myl],Pyl,"tyl",…,
[nyl myl],Pyl,"tyl")
```

```
[yl,…,yk]=call("ident",xl,…,xn)
```

```
→ C=call("matmul",A,1,"d",n,2,"i",m,3,"i",B,4,"d",l,5,"i","out",[n,l],6,"d");
```

第 6 章　SCILAB 的应用举例

6.1　引言

6.2　在求解线性方程组方面的应用

$$
\begin{aligned}
&i_1R_1+(i_1+i_2)R_2+(i_1+i_3)R_3=V_1\\
&(i_1+i_2)R_2+(i_2-i_3)R_5=V_2\\
&(i_1+i_3)R_3+i_3R_4+(i_3-i_2)R_5=0
\end{aligned}
$$

$$
\begin{bmatrix}
R_1+R_2+R_3 & R_2 & R_3\\
R_2 & R_2+R_5 & -R_5\\
R_3 & -R_5 & R_3+R_4+R_5
\end{bmatrix}
\begin{bmatrix}i_1\\ i_2\\ i_3\end{bmatrix}=
\begin{bmatrix}V_1\\ V_2\\ V_3\end{bmatrix}
$$

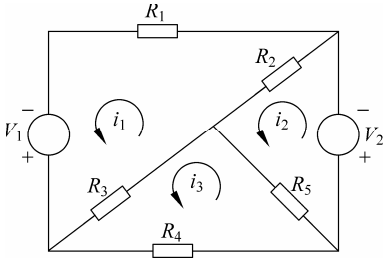

图 6.1　例 6.1 图

```
→ R=[2  29  8  15  4];
→ b=[10  25  0]';
→ A=[R(1)+R(2)+R(3)    R(2)          R(3); …
       R(2)            R(2)+R(5)   -R(5); …
       R(3)             +R(5)      R(3)+R(4)+R(5)];
→ [x KerA]=linsolve(A,-b);
→ x=
   !  -1.5604027 !

   !  2.2248322   !
   !  0.7919346   !
→ KerA=
     [  ]
```

$$HNO_3+Cu\Rightarrow Cu(NO_3)_2+H_2O+NO_2$$

$$\begin{pmatrix}0\\1\\1\\3\end{pmatrix},\quad \begin{pmatrix}1\\0\\0\\0\end{pmatrix},\quad \begin{pmatrix}1\\0\\2\\6\end{pmatrix},\quad \begin{pmatrix}0\\2\\0\\1\end{pmatrix},\quad \begin{pmatrix}0\\0\\1\\2\end{pmatrix}$$

$$\begin{pmatrix}0\\1\\1\\3\end{pmatrix}x_1+\begin{pmatrix}1\\0\\0\\0\end{pmatrix}x_2=\begin{pmatrix}1\\0\\2\\6\end{pmatrix}x_3+\begin{pmatrix}0\\2\\0\\1\end{pmatrix}x_4+\begin{pmatrix}0\\0\\1\\2\end{pmatrix}x_5$$

$$\begin{bmatrix}0&1&-1&0&0\\1&0&0&-2&0\\1&0&-2&0&-1\\3&0&-6&-1&-2\end{bmatrix}\begin{pmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{pmatrix}=\begin{pmatrix}0\\0\\0\\0\end{pmatrix}$$

```
→ A=[0  1  -1  0  0;  1  0  0  -2  0;  1  0  -2  0  -1;  3  0  -6  -1  -2];
→ b=[0  0  0  0]';
→ [x,  KerA]=linsolve(A,b);
→ z=KerA(1);
→ for i=2:5
→     if(z)>KerA(i)
→         z=KerA(i);
→     end
→ end
→ q=1;
→ r=1;
→ while(r==1)
→     r=0;
→     c=q*KerA/z;
→     for i=1:5
→         if(c(i)-int32(c(i)))>0.1
→           r=1;
→           q=q+1;
→         end
→         if r==1
→         break
→         end
→     end
→ end
→ disp(c);
    c=
    ! 4 !
    ! 1 !
    ! 1 !
    ! 2 !
    ! 2 !
```

表 6.1　玉米植株高度与生长阶段的关系

生长阶段	1	2	3	4	5	6	7	8	9	10	11
植株高度/cm	0.67	0.85	1.28	1.75	2.27	2.75	3.69	4.71	6.36	7.73	9.91
生长阶段	12	13	14	15	16	17	18	19	20	21	22
植株高度/cm	12.75	16.55	20.1	27.35	32.55	37.55	44.75	53.38	71.61	83.89	97.46
生长阶段	23	24	25	26	27	28	29	30	31		
植株高度/cm	112.73	135.12	153.6	160.32	167.05	174.9	177.87	180.19	180.79		

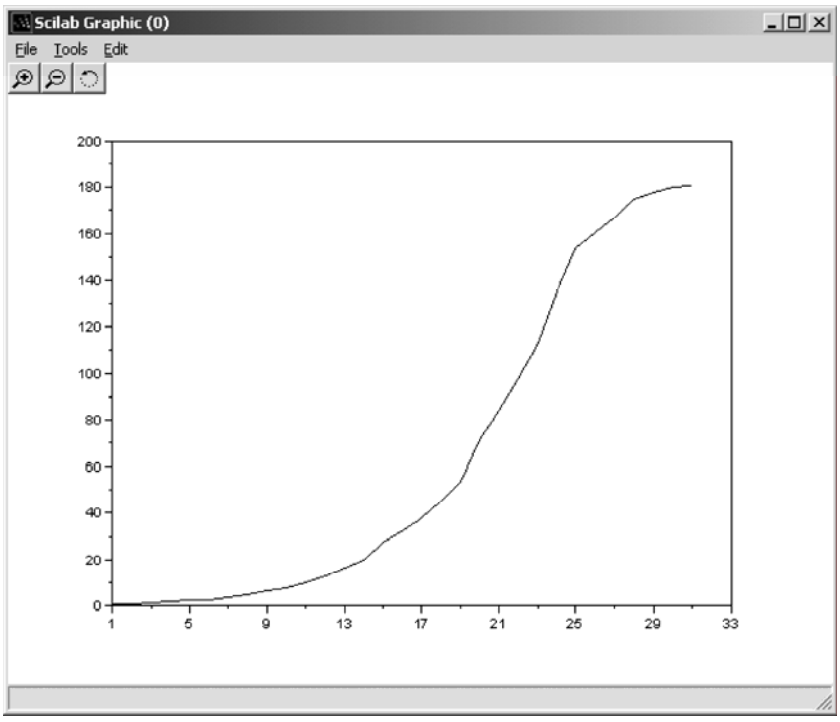

图 6.2　玉米植株与生长阶段关系

$$y = ae^{bt}$$

$$y = 0.534e^{0.233t}$$

表 6.2　根据经验公式 $y=0.534e^{0.233t}$ 计算出的玉米植株高度与生长阶段的关系

生长阶段	1	2	3	4	5	6	7	8	9	10	11	12	13
函数值 $y(t)$	0.67	0.85	1.07	1.36	1.71	2.16	2.73	3.44	4.34	5.48	6.92	8.74	11.03
生长阶段	14	15	16	17	18	19	20	21	22	23	24	25	
函数值 $y(t)$	13.93	17.58	22.2	28.02	35.37	44.66	56.37	71.16	89.84	113.41	143.17	180.73	

$$\begin{cases} ae^{2b} = 0.85 \\ ae^{25b} = 112.73 \end{cases}$$

$$y = ae^{bt}$$

$$\ln y = \ln a + bt$$

$$\ln y = c + bt$$

$$\ln y = c + bt$$

$$\ln y_i = c + bt_i \qquad i = 1,2,\cdots,25$$

```
→ alpha = ones(1,25);
→ bata = [1:25];
→ f = [0.67  0.85  1.28  1.75  2.27  2.75  3.69  4.71  6.36  7.73  9.91  12.75  16.55
20.1  27.35  32.55  37.55  44.75  53.58  71.61  83.89  97.46  112.73  135.12  153.6];
→ bl = log(f)
→ al = [alpha; bata]';
→ x = pinv(al) * bl;          // pinv(al)表 al 的广义逆,即为(A^T A)^-1 A^T
→ a = exp(x(1));
→ b = x(2);
```

$$y = ae^{bt}$$

6.3　在求解非线性方程（组）方面的应用

```
[x [,v [,info]]] = fsolve(x0,fct [,fjac][,tol])
```

$$\begin{cases} x^2 + y^2 - 1 = 0 \\ x - y + c = 0 \end{cases}$$

```
→ for i = 1:101
→ x(i) = (i - 51) / 50;
→ y(i) = sqrt(1 - x(i)^2);
→ z(i) = - y(i);
→ end
→ l = x + 2;
→ plot2d(x,[y,z,l], logflag = "nn", frameflag = 3, rect = [-2,-2,2,2]);
```

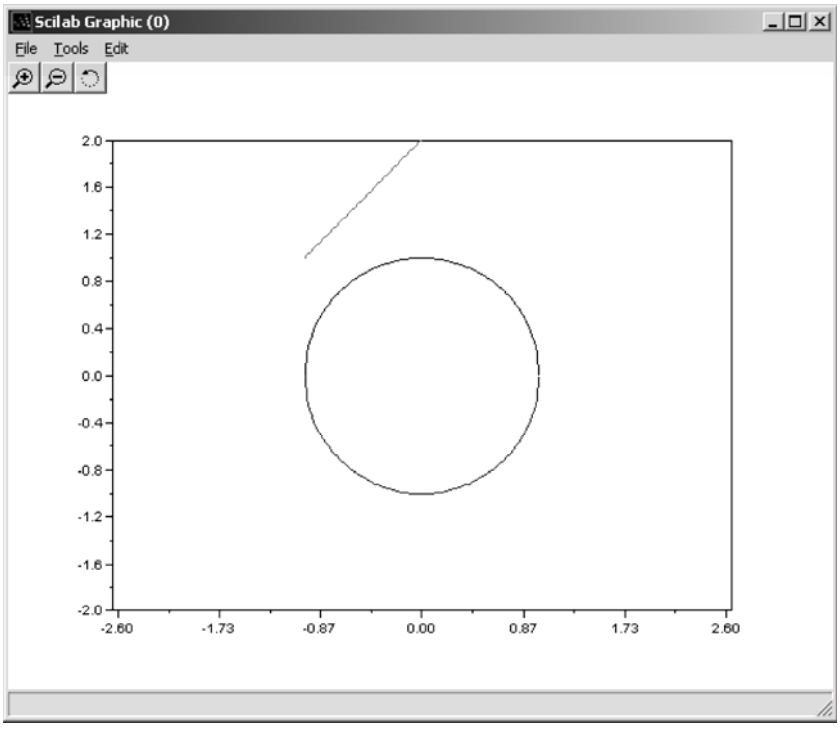

图 6.3　直线与圆相离情形（方程组无解）

```
→ l = x + sqrt(2);
→ plot2d(x,[y,z,l], logflag = "nn", frameflag = 3, rect = [ - 2, - 2,2,2]);
```

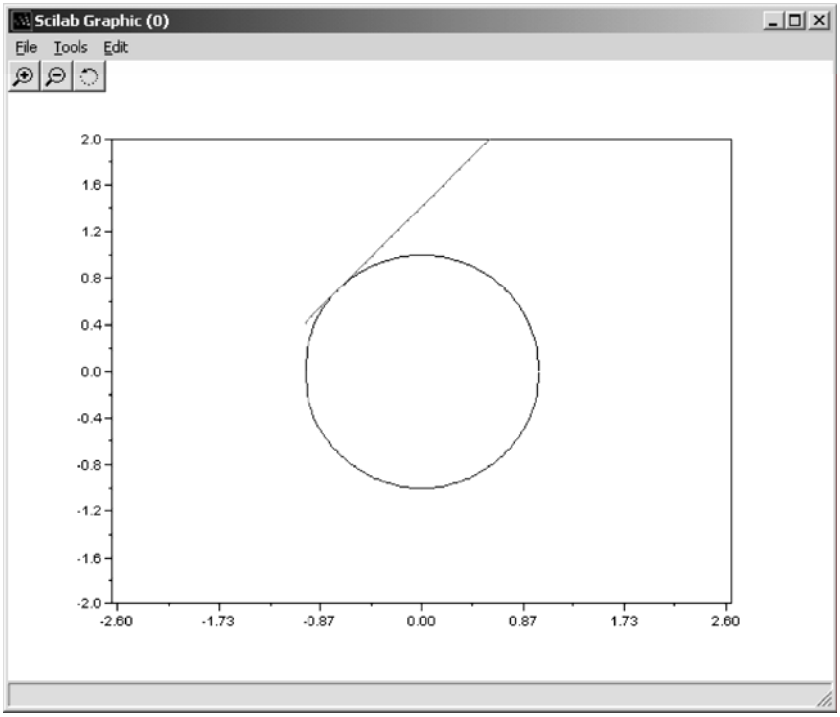

图 6.4　直线与圆相切情形（方程组有一解）

```
l = x + 1;
plot2d(x,[y,z,l], logflag = "nn", frameflag = 3, rect = [ - 2, - 2,2,2]);
```

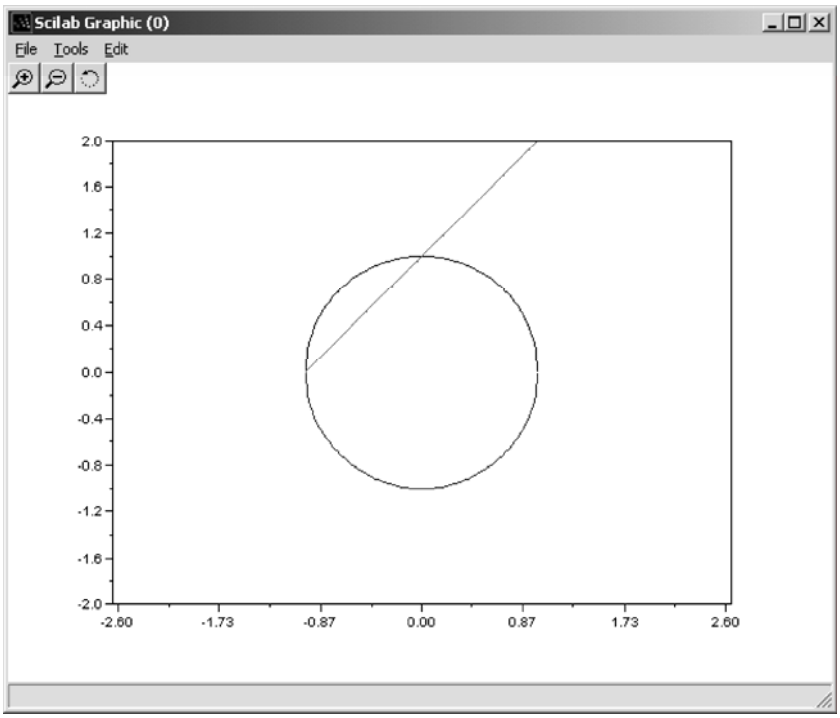

图 6.5　直线与圆相交于两点情形（方程组有二解）

$$\begin{cases} x^2+y^2-1=0 \\ x-y+c=0 \end{cases}$$

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}\begin{pmatrix} x^2 \\ y^2 \end{pmatrix}+\begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}\begin{pmatrix} x \\ y \end{pmatrix}+\begin{bmatrix} -1 \\ c \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{a}=\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix},\quad \boldsymbol{r}=\begin{bmatrix} -1 \\ c \end{bmatrix},(\text{其中 } c \text{ 待定})$$

$$x\hat{}2=\begin{pmatrix} x^2 \\ y^2 \end{pmatrix},\quad x=\begin{pmatrix} x \\ y \end{pmatrix}$$

$$ax\hat{}2+bx+r=0$$

```
a=[1  1;  0  0];
```

```
b=[0  0;  1  -1];
r=[-1; 2]     // 此处 c—2,以后逐次改为 sqrt(2),1 等
deff('[y]=f(x)',y=a * x^2+b * x+r);
[x1,f1,iN1]=fsolve([2; 2],f);          // 根据迭代初值[2; 2]求解方程,x1 为方程组的
    // "解",f1 为方程左端函数在 x1 处的值,凭此能够判断 x1 是否为方程组的解,iN 1 可
    // 取 0,1,2,3,4 等值,据此可知迭代是否收敛等
```

```
[x1,f1,iN1]=fsolve([2; 2],f)
```

$$\begin{cases} ae^{2b}=0.85 \\ ae^{25b}=112.73 \end{cases}$$

$$\begin{aligned} &f1(a,b)=ae^{2b}-0.85 \\ &f2(a,b)=ae^{23b}-112.73 \end{aligned}$$

$$\begin{bmatrix} e^{2b} & 2ae^{2b} \\ e^{23b} & 23ae^{23b} \end{bmatrix}$$

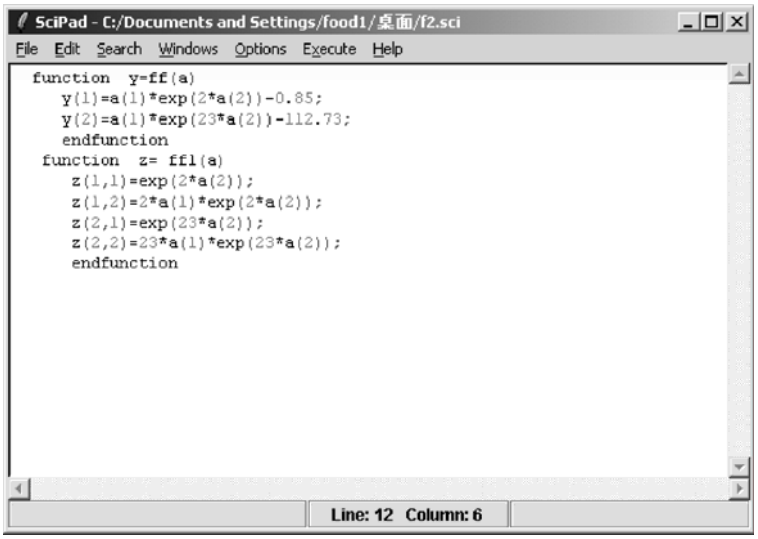

图 6.6　在 SCILAB 编辑器内编写的函数

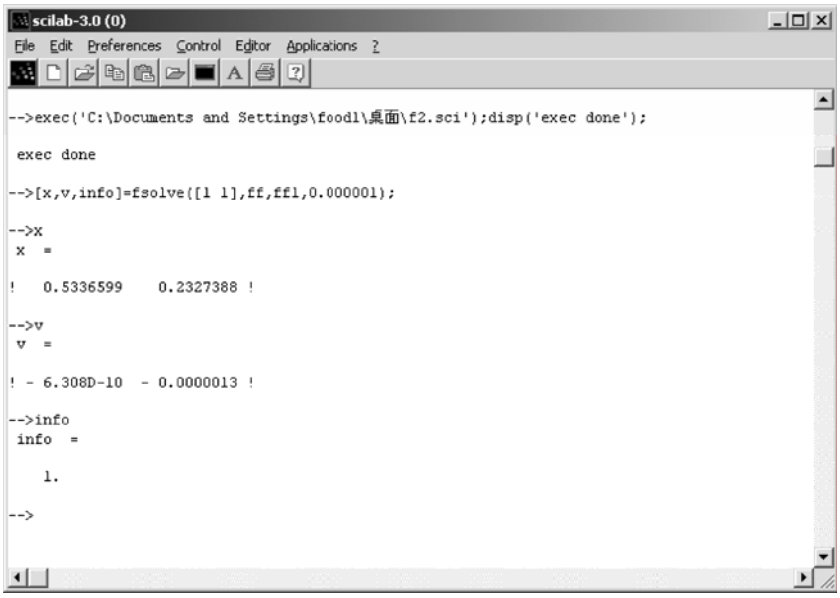

图 6.7　在 SCILAB 主窗口调用 fsolve

6.4　SCILAB 在函数插值方面的应用

```
d = splin(x,y[,spline-type[,der]])
```

$$s^{(3)}(x_2-) = s^{(3)}(x_2+), \quad s^{(3)}(x_{n-1}-) = s^{(3)}(x_{n-1}+)$$

$$s'(x_1) = \mathrm{der}(1), \quad s'(x_n) = \mathrm{der}(2)$$

$$s''(x_0) = 0, \quad s''(x_n) = 0$$

$$s'(x_1) = s'(x_n), \quad s''(x_1) = s''(x_n)$$

```
[der(1)  der(2)] = [y'(x1)  y'(xn)]    (其中 y(x)为被插函数)
```

```
[yp[,yp1[,yp2[,yp3]]]] = interp(xp,x,y,d[,out-mode])
```

```
→ x = 1:31;
→ y = [0.67  0.85  1.28  1.75  2.27  2.75  3.69  4.71  6.36  7.73  9.91  12.75
    16.55  20.1  27.35  32.55  37.55  44.75  53.58  71.61  83.89  97.46  112.73
    135.12  153.6]';
→ d = splin(x,y,"clamped",[0.18 0.60]);
→ xx = [1.5  3.47]; //依此插值方法构造的经验公式求在 t=1.5 和 3.47 处的值
→ yy = interp(xx,x,y,d);
→ xx
  xx =
      ! 1.5  3.47 !
→ yy
  yy =
      ! 0.7457183  1.4961656 !
```

6.5 意犹未尽的话

附录　SCILAB 部分函数指令表

1. 通用指令			
help	在线帮助	|,or	逻辑或
apropos	文档中关键词搜寻	~,not	逻辑非
ans	缺省变量名以及最新表达式的运算结果	:	冒号
clear	从内存中清除变量和函数	( )	圆括号
exit	关闭 SCILAB	[]	方括号
quit	退出 SCILAB	{}	花括号
save	把内存变量存入磁盘	.	小数点
exec	运行脚本文件	,	逗号
mode	文件运行的显示格式	;	分号
getversion	显示 SCILAB 版本	//	注释号
ieee	浮点运算溢出显示模式选择	=	赋值符号
who	列出工作内存中的变量名	'	引号
edit	文件编辑器	.'	复数转置号
type	变量类型	ans	转置号
what	列出 SCILAB 基本指令	%eps	最新表达式的运算结果
format	设置数据输出格式	%i	浮点误差容限 $-2^{-51}\approx 2.22\times 10^{-16}$
chdir	改变当前工作目录	%inf	虚数单位 $-\sqrt{(-1)}$
getenv	给出环境值	%pi	正无穷大
mkdir	创建目录	3. 编程语言结构	圆周率，π=3.1415926535897…
pwd	显示当前工作目录	abort	中止计算或循环
evstr	执行表达式	break	终止最内循环
2. 运算符和特殊算符		case	同 select 一起使用
+	加	continue	将控制转交给外层的 for 或 while 循环
-	减	else	同 if 一起使用
*	矩阵乘	elseif	同 if 一起使用
.*	数组乘	end	结束 for，while，if 语句
^	矩阵乘方	for	按规定次数重复执行语句
.^	数组乘方	if	条件执行语句
\	反斜杠或左除	otherwise	可同 switch 一起使用
/	斜杠或右除	pause	暂停模式
./或.\	数组除	return	返回
==	等号	select	多个条件分支
~=	不等号	then	同 if 一起使用
<	小于	while	不确定次数重复执行语句

续表

>	大于	eval	特定值计算
<=	小于或等于	feval	函数特定值计算或多变量计算
>=	大于或等于	function	函数文件头
&.,and	逻辑与	global	定义全局变量
isglobal	检测变量是否为全局变量	interp	插值函数
error	显示错误信息	interpln	线性插值函数
lasterror	显示最近的错误信息	intsplin	样条插值函数
sprintf	按格式把数字转换为串	smooth	样条平滑函数
warning	显示警告信息	spline	样条函数
4.基本数学函数		quarewave	方波函数
acos	反余弦	sign	符号函数
acosh	反双曲余弦	double	将整数转换为双精度浮点数
acot	反余切	5.基本矩阵函数和操作	
acoth	反双曲余切	eye	单位阵
acsc	反余割	zeros	全 0 矩阵
acsch	反双曲余割	ones	全 1 矩阵
asin	反正弦	rand	均匀分布随机阵
asinh	反双曲正弦	genmarkov	生成随机 Markov 矩阵
atan	反正切	linspace	线性等分向量
atanh	反双曲正切	logspace	对数等分向量
cos	余弦	logm	矩阵对数运算
cosh	双曲余弦	cumprod	矩阵元素累计乘
cotg	余切	cumsum	矩阵元素累计和
coth	双曲余切	toeplitz	Toeplitz 矩阵
sin	正弦	disp	显示矩阵和文字内容
sinh	双曲正弦	length	确定向量的长度
tan	正切	sum	元素和
tanh	双曲正切	trapz	梯形数值积分
exp	指数	corr	求相关系数或方差
log	自然指数	6.稀疏矩阵运算	
log10	常用对数	sparse	稀疏矩阵(只存储非零元素)
log2	以 2 为底的对数	adj2sp	邻接矩阵转换为稀疏矩阵
sqrt	平方根	full	稀疏矩阵转换为全矩阵
abs	绝对值	Mtlb_sparse	将 SCILAB 稀疏矩阵转换为 MATLAB 稀疏矩阵格式
conj	复数共轭	sp2adj	将稀疏矩阵转换为邻接矩阵
imag	复数虚部	speye	稀疏矩阵方式单位阵
real	复数实部	sprand	稀疏矩阵方式随机矩阵
ceil	向上(正无穷大方向)取整	spzeros	稀疏矩阵方式全零阵
fix	向零方向取整	lufact	稀疏矩阵 LU 分解
floor	向下(负无穷大方向)取整	lusolve	稀疏矩阵方程求解
round	四舍五入取整	spchol	稀疏矩阵 Cholesky 分解
sign	符号函数	7.输入输出函数	

续表

gsort	降次排序	diry	生成屏幕文本记录
erf	误差函数	disp	变量显示
erfc	补误差函数	file	文件管理
gamma	gamma 函数	input	用户键盘输入
load	读已存的变量	strsubst	字符串中的字符替换
mclose	关闭文件	10. 日期与时间	
mget	读二进制文件	date	日期
mgetl	按行读 ASCII 码文件	getdate	读日期与时间
mgetstr	读字符串中单个字	timer	CPU 时间计时
mopen	打开文件	11. 二维图形函数	
mut	写二进制文件	plot2d	直角坐标下线性刻度曲线
mfscanf	读 ASCII 码文件	champ	二维向量场
print	将变量记录为文件	champl	由颜色箭头表示的二维向量场
read	读矩阵变量	contour2d	等高线图
save	将变量存为二进制文件	errbar	曲线上增加误差范围框线条
strartup	启动文件	grayplot	应用颜色表示的表面
write	按格式存文件	xgrid	画坐标网格线
xgetfile	对话方式获取文件路径	histplot	统计频数值方图
x_dialog	建立 Xwindows 参数输入对话框	Matplot	散点图阵列
Tk_Getvar	得到 Tk 文件变量	12. 三维图形函数	
Tk_EvalFile	执行 Tk 文件	plot3d	三维表面
8. 函数与函数库操作		plot3d1	用颜色或灰度表示的三维表面
deff	在线定义函数	param3d	三维中单曲线
edit	函数编辑器	param3d1	三维中多曲线
function	打开函数定义	contour	三维表面上的等高线图
functions	SCILAB 函数或对象	hist3d	三维表示的统计频数直方图
genlib	在给定目录下建立所有文件的函数库	geom3d	三维向二维上的投影
get_function_path	读函数库的文件存储目录路径	13. 线条类图形	
		xpoly	单线条或单多边形
getd	读函数库中的全部文件	xpolys	多线条或多边形
getf	在文件中定义一个函数	xrpoly	正多边形
lib	函数库定义	xsegs	非连接线段
macro	SCILAB 函数或对象	xfpoly	单个多边形内填充
macrovar	输入变量个数	xfpolys	多个多边形内填充
newfun	输出变量个数	xrect	矩形
9. 字符串操作		xfrect	单个矩形内填充
code2str	将 SCILAB 数码转换为字符串	xrects	多个矩形内填充
convstr	字母大小转换	xarc	单个弧线段或弧圈
emptystr	清空字符串	xarcs	多个弧线段或弧圈
grep	搜寻相同字符串	xfarc	单个弧线段或弧圈填充
part	字符提取	xfarcs	多个弧线段或弧圈填充
str2code	将字符串转换为 SCILAB 数码	xarrows	多箭头
string	字符串转换	14. 图形注释,变换	

续表

strings	SCILAB 对象,字符串	xstring	图形中字符
strcat	连接字符	xstringb	框内字符
strindex	字符串的字符位置搜寻	xtitle	图形标题
xaxis	轴名标注		
plotframe	图形加框并画坐标网格线	locate	由单击读入图形中的多点位置坐标
isoview	等尺寸比例显示(原图形窗口不改变)	xgetmouse	由单击读入图形中的当前点位置坐标
square	等尺寸比例显示(原图形窗口改变)	18.系统控制	
xsetech	设置小窗口	abcd	状态空间矩阵
xchange	转换实数为图形像素坐标值	cont_mat	可控矩阵
subplot	设置多个子窗口	csim	线性系统时域响应
15.图形颜色及图形文字		dsimul	状态空间的离散时域响应
colormap	应用颜色图	feedback	反馈操作符
getcolor	交互式选择颜色图	flts	时域响应(离散、采样系统)
addcolor	增加新色于颜色图	frep2tf	基于传递函数的频域响应
graycolormap	线性灰度图	freq	频域响应
hotcolormap	热色(红到黄色)颜色图	g_margin	幅值裕量
xset	图形显示方式设定	imrep2ss	基于状态空间的脉冲响应
xget	读当前图形显示方式设定	lin	线性化操作
getsymbol	交互式选择符号和尺寸	lqe	Kalman 滤波器
16.图形文件及图形文字		lqg	LQG 补偿器
xsave	将图形存储为文件	lqr	LQ 补偿器
xload	从磁盘中读出图形文件	ltitr	基于状态空间的离散时域响应
xbasimp	将图形按 PS 文件打印或存储为文件	obscont	基于观测器的控制器
xs2fig	将图形生成 Xfig 格式文件	observer	观测器
xbasc	取消图形及其相关内容	obsv_mat	观测矩阵
xclear	清空图形窗	p_margin	相位裕量
driver	选择图形驱动器	phasemag	相位与幅值计算
xinit	图形驱动器初始化	ppol	极点配置
xend	关闭图形	repfreq	频域响应
xbasr	图形刷新	ricc	Riccati 方程
replot	更改显示范围后的图形刷新	ftitr	基于传递函数的离散时域响应
xdel	关闭图形	sm2ss	系统矩阵到状态空间的变换
xname	改变当前图形窗名称	ss2ss	反馈连接的状态空间到状态空间的变换
17.控制分析用图形		ss2tf	状态空间到传递函数的变换
bode	波特图坐标	stabil	稳定性计算
gainplot	幅值图坐标(波特图中的幅值图)	tf2ss	传递函数到状态空间的变换
nyquist	奈奎斯特图	time_id	SISO 系统最小方差辨识
m_circle	M-圆图	19.鲁棒控制	
chart	尼库拉斯图	augment	被控对象增广操作
black	Black-图	bstap	Hankel 矩阵近似
evans	根轨迹图	ccontrg	H_∞ 控制器
sgrid	s 平面图	dhnorm	离散 H_∞ 范数
plzr	零极点图	h2norm	H_1 范数

续表

zgrid	z平面图	h_cl	闭环矩阵
20. 图形应用中的其他指令		h_inf	H_∞控制器
graphics	图形库指令表	h_norm	H_∞范数
xclick	等待鼠标在图形上的单击输入	hankelsv	Hankel矩阵奇异值
leqr	H_∞控制器的LQ增益	ode	常微分方程
linf	无穷范数	ode_discrete	离散常微分方程
riccati	Riccati矩阵	ode_root	常微分方程根解
sensi	敏感函数	odedc	连续离散常微分方程
21.动态系统		optim	非线性优化
arma	ARMA模型	quapro	线性二次型规划
arma2p	基于AR模型获得多项式矩阵	semidef	半正定规划
armac	RMAX辨识	24.多项式计算	
arsimul	RMAX系统仿真	coeff	多项式系数
noisegen	噪声信号发生器	coffg	多项式矩阵逆
odedi	常微分方程仿真检测	degree	多项式阶数
prbs_a	伪随机二进制序列发生器	denom	分母项
reglin	线性拟合	derivat	有理矩阵求导
22.系统与控制实例		determ	矩阵的列式值
artest	Arnold动态系统	factors	因式分解
bifish	鱼群人口发展的离散时域模型	hermit	Hermit型
boucle	具有观测器的动态系统相位图	horner	多项式计算
chaintest	生物链模型	invr	有理矩阵逆
gpech	渔业模型	lcm	最小公倍数
fusee	登陆火箭问题	ldiv	多项式矩阵长除
lotest	Lorennz吸引子	numer	分子项
mine	采矿问题	pdiv	多项式矩阵除
obscontl	可控可观系统	pol2des	将多项式矩阵变换为表达式
portr3d	三维相位图	pol2str	将多项式变换为字符串
protrait	二维相位图	polfact	最小因式
recur	双线性回归方程	residu	余量
systems	动态系统	roots	多项式的根
tangent	动态系统的线性化	simp	多项式的简化
tadinit	动态系统的交互初始化	systmat	系统矩阵
23.非线性工具(优化与仿真)		25.信号处理	
bvode	边界值问题的常微分方程	%asn	椭圆积分
dsrt	隐式微分方程过零解	%k	Jacobi完全椭圆积分
dassl	代数微分方程	%sn	Jacobi椭圆函数
datafit	基于测量数据的参数辨识	analpf	模拟量低通滤波器
derivative	导数计算	buttmag	Butterworth滤波器响应
fsolve	非线性函数过零解	cepstrum	倒谱计算
impl	线性微分方程	cheb1mag	Chebyshev一型响应
int2d	二维定积分	cheb2mag	Chebyshev二型响应
int3d	三维定积分	chepol	Chebyshev多项式

续表

intg	不定积分	convol	卷积
leastsq	非线性最小二乘法	corr	相关,协方差
linpro	线性规划	cspect	谱估计(应用相关法)
lmisolver	线性不等矩阵	dft	离散傅里叶变换
fft	快速傅里叶变换	auwrite	写 *.au 音频文件
filter	滤波器建模	lin2mu	将线性信号转换为 μ 率码信号
fsfirlin	FIR 滤波器设计	loadwave	取 *.wav 音频文件
hank	协方差矩阵到 Hankel 矩阵变换	mapsound	音频信号图示
hilb	Hilbert 变换	mu2lin	将 μ 率码信号转换为线性信号
iir	IIR 数字滤波器	playsnd	音频信号播放
intdec	信号采样率更改	savewave	存 *.wav 音频文件
kalm	Kalman 滤波器更新	wavread	读 *.wav 音频文件
mese	最大熵谱估计	wavwrite	写 *.wav 音频文件
mfft	多维快速傅里叶变换	27.语言与数据转换工具	
mrfit	频率响应拟合	ascii	字符串的 ASCII 码
phc	Markov 过程	excel2sci	读 ASCII 格式的 Excel 文件
srkf	Kalman 滤波器平方根	fun2string	将 SCILAB 函数生成 ASCII 码
sskf	稳态 Kalman 滤波器	mfile2sci	将 MATLAB 的 M 格式文件转换为 SCI 格式文件
system	观测更新		
wfir	线性相位 FIR 滤波器	mtlb_load	取 MATLAB 第 4 版本文件中变量
weiener	Weiener(维纳)滤波器	matlb_save	按 MATLAB 第 4 版本文件格式存变量
window	对称窗函数	pol2tex	将多项式转换为 TeX 格式
yulewalk	最小二乘滤波器	sci2for	将 SCILAB 函数转换为 FORTRAN 格式文件
zpbutt	Buthererworth 模拟滤波器		
zpchl	Chebyshev 模拟滤波器	texprint	按 TeX 格式输出 SCILAB 对象
26.音频信号		translatepaths	将子目录下的所有 MATLAB 文件转换为 SCI 文件格式
analyze	音频信号频域图		
auread	读 *.au 音频文件		

(注:本指令表只收集了部分常用指令,有关全部指令请参照文档文件)

参考文献